PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :
G06F 15/20, G09G 1/00

A1

(11) International Publication Number: WO 92/16898

(43) International Publication Date: 1 October 1992 (01.10.92)

(21) International Application Number: PCT/US91/01652

(22) International Filing Date: 12 March 1991 (12.03.91)

(71) Applicant *(for all designated States except US)*: WANG LABORATORIES, INC. [US/US]; One Industrial Avenue, Lowell, MA 01851 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)* : KNOWLTON, Kenneth, C. [US/US]; 51 Pond View Drive, Merrimack, NH 03045 (US).

(74) Agents: SHANAHAN, Michael, H. et al.; One Industrial Avenue, Lowell, MA 01851 (US).

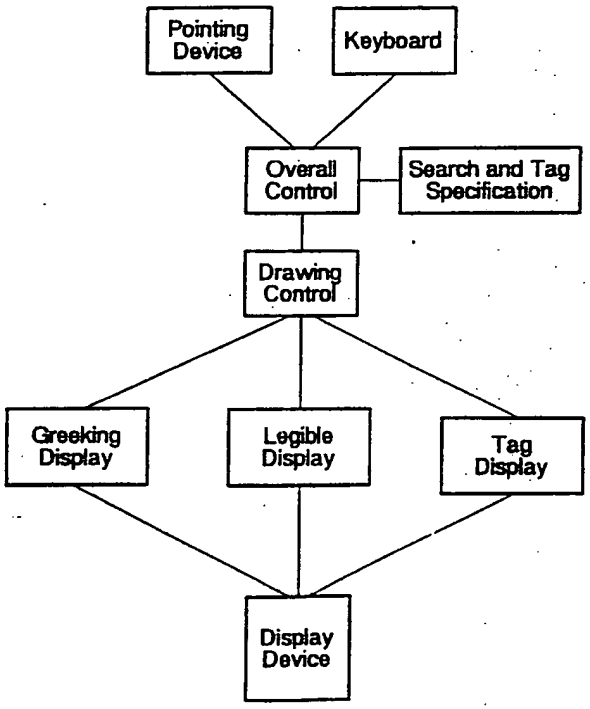
(81) Designated States: AT (European patent), AU, BE (European patent), CA, CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), GR (European patent), IT (European patent), JP, LU (European patent), NL (European patent), SE (European patent), US.

Published
With international search report.

(54) Title: DISPLAY SYSTEM AND METHOD FOR CONTEXT-BASED SCROLLING

(57) Abstract

A computer-based system for displaying all, or at least many pages of, a document, and including means for scrolling through the document for purposes such as browsing or editing. Most of the display shows text in a "greeked" form with one line of pixels representing one line of text. The greeked form, although illegible, shows shapes of text in the document by indicating the presence or absence of text. Two windows of legible text appear as if two bar-type magnifying glasses were placed over the illegible greeked text. A window may be moved to a new location by touching a stylus to a digitizing tablet (or by clicking a mouse) or by dragging with the stylus (or mouse). When the two windows are adjacent, they become latched and can be moved as one large window. Words of special significance (e.g., embedded labels or search results) can be displayed as legible tags wherever they appear, in the greeked text as well as in the legible text. The greeking is arranged to produce a checkerboard of pixels, that appears as gray, permitting black tags to stand out clearly. Because the tags appear throughout the display at locations corresponding to their locations in the document, they can add significantly to the context presented by the greeking, permitting the user to become oriented and to locate desired portions of the displayed document. Further, tags can permit a user to quickly ascertain the locations and significance of search results.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FI	Finland	ML	Mali
AU	Australia	FR	France	MN	Mongolia
BB	Barbados	GA	Gabon	MR	Mauritania
BE	Belgium	GB	United Kingdom	MW	Malawi
BF	Burkina Faso	GN	Guinea	NL	Netherlands
BG	Bulgaria	GR	Greece	NO	Norway
BJ	Benin	HU	Hungary	PL	Poland
BR	Brazil	IE	Ireland	RO	Romania
CA	Canada	IT	Italy	RU	Russian Federation
CF	Central African Republic	JP	Japan	SD	Sudan
CG	Congo	KP	Democratic People's Republic of Korea	SE	Sweden
CH	Switzerland			SN	Senegal
CI	Côte d'Ivoire	KR	Republic of Korea	SU	Soviet Union
CM	Cameroon	LI	Liechtenstein	TD	Chad
CS	Czechoslovakia	LK	Sri Lanka	TG	Togo
DE	Germany	LU	Luxembourg	US	United States of America
DK	Denmark	MC	Monaco		
ES	Spain	MG	Madagascar		

DISPLAY SYSTEM AND METHOD FOR CONTEXT-BASED SCROLLING

FIELD OF THE INVENTION

The present invention relates to computer-based systems and methods for displaying, navigating through, and editing documents.

BACKGROUND

It is common for computer-based systems to provide for viewing of a document that is larger than can legibly be displayed to the user at one time. In most cases, the document is conceived as a linear, or two-dimensional, arrangement of text or other data, a portion of the document is displayed, and commands are made available to the user to move the display forward and backward through the document. For example, the system may be provided with a screen forward key and a screen backward key, thereby permitting the user to move forward to the next or previous displayable portion of the document. Commands may also be provided to move by other size units of the document, such as by line of text or by document page (which may differ in size from the amount that can be displayed at one time).

Since the user only sees a portion of the document the user may have no way of determining what portion of the overall document is being viewed, without starting at one end of the document and stepping through the document a screen at a time. The user lacks a context for the portion being displayed. Moreover, the user has no convenient way of going directly to a part of the document that might be recognizable by its coarse visible features and by its position in the overall document.

To address the problem of lack of context, some systems provide a numeric indication of where in the overall document the portion that is currently being

displayed (or, more precisely, the current cursor location) is located, e.g., page number, line number, percentage of the total document length.

Since approximately the middle of the 1970s (e.g., work done at Xerox Corporation's Palo Alto Research Center), it has become increasingly common for user interfaces to provide for navigation through a document by means of a scroll bar. A scroll bar includes a marker that provides an analog visual indication of the location of the currently displayed portion. For example, Microsoft Corporation has produced various versions of a product known as Windows, which uses scroll bars. A user positions a mouse pointer at various locations on a scroll bar and clicks or drags to move through the document. Fig. 1 shows the display of a portion of a document and shows both horizontal and vertical scroll bars for use in navigating through a document, as provided in the Windows "notepad" application.

The vertical scroll bar includes several elements: at the top of the bar is an up arrow; at the bottom of the bar is a down arrow; in the region between the arrows is positioned a movable block. The relative position of the block indicates the approximate relative location within the document of the portion that is displayed. In the notepad application, the scroll bar operates as follows: clicking on the up arrow moves the document down one line (i.e., as if the window into the document is moving up); clicking on the down arrow moves the document up one line; clicking in the region above the movable block moves the document down one screen; clicking in the region below the movable block moves the document up one screen; dragging the movable block causes the display to scroll to the position within the document indicated by the relative position of the movable block within the scroll bar.

SUMMARY OF THE INVENTION

A system according to the present invention displays the context of a much larger portion of a document than can be legibly presented in the available display area. This is accomplished by presenting the document in a "greeked" form, which is visually compressed to such a degree that the characters of the text are not legible, but larger shapes remain visible — i.e., overall shapes in the document, such as headings, blank lines, paragraphs, and indentations. Windows of legible

text are movable through the greeking, as if magnifiers were moved across a distant or miniature document.

Furthermore, legible tags are displayed at appropriate locations in the document in the greeked area, as well as in the legible portion. For example, when the tags show the result of a search for all occurrences of a character string, the tags allow a user to refer quickly to all of the occurrences of the string (such as to see where a subroutine is defined and all the places where it is called).

Highlighting such as by color can also be used to identify significant portions of the document in both the legible windows and in the greeked portions.

Preferably, at least a portion of the greeked text is compressed in only the vertical dimension. In this case, the greeking is of the same width as the legible portion, permitting a smooth visual transition from one to the other.

The present inventive display arrangement is suitable for navigation through more than just "documents" in a narrow sense. It suitable for moving through large bodies of other displayable material, such as data retrieval results.

Although useful as a stand-alone browser, this display arrangement is suitable for incorporation into a variety of computer-based applications requiring display and possibly modification of more than can fit in the area available for display.

BRIEF DESCRIPTION OF THE DRAWING

The invention is pointed out with particularity in the appended claims. The above and other advantages of the invention may be better understood by referring to the following detailed description in conjunction with the drawing, in which:

Fig. 1 shows prior art screen display that includes scroll bars.

Fig. 2 is a block diagram illustrating physical aspects of an illustrative system implementing the present invention.

Fig. 3 is a block diagram illustrating logical aspects of an illustrative system implementing the present invention.

Fig. 4 shows a presentation on a display screen according to the present invention, with a single window of legible text.

Fig. 5 shows a display similar to Fig. 4, but with two windows and with tags indicating the results of a search.

Fig. 6 shows a portion of the display of Fig. 5 on an expanded scale, such that individual pixels are visible.

Fig. 7 shows a display where embedded tags have been made visible.

Fig. 8 shows an portion of the display of Fig. 7 on an expanded scale.

Figs. 9, 10, 11, and 12 each show the same portion of the display as shown in Fig. 8 with the window of enlarged text moved down one line in each successive figure.

Fig. 13 is an overall flowchart of a program used in implementing the present invention.

Figs. 14 and 15 are flowcharts detailing portions of the flowchart of Fig. 13; Fig. 14 details keystroke processing, and Fig. 15 details stylus processing and window movement.

Fig. 16 illustrates schematically a screen display according to an embodiment of the invention particularly suited to longer documents.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

An illustrative embodiment of the present invention (referred to below as "the scroll system") is implemented with a computer system and a computer program (referred to below as SCROLL).

Fig. 2 is a block diagram illustrating a physical arrangement of the scroll system. The computer system in this illustrative embodiment includes a processor such as the 80386 (Intel Corporation) running the MSDOS operating system (Microsoft Corporation), a high resolution display having a display area of 1024 by 1024 binary pixels, and a digitizing tablet and associated stylus. The computer system

includes memory (both mass storage and main memory) in which SCROLL (as well as other programs and data) is stored. The invention could readily be implemented using other computer systems: other processors, other operating systems, other displays (e.g., VGA), and other pointing devices (e.g., mouse) could be used. Given the graphical nature of the displays produced by the invention, one skilled in the art will appreciate the implications of processor and display controller performance on the utility of the invention. For example, a system with less speed at performing the graphical operations required will appear to a user to be less responsive to the user's actions. Also, considering the manner in which the display is utilized, one skilled in the art will appreciate the implications of displays of various sizes, bit-plane depths, and resolutions.

The SCROLL program is written in the C programming language, and this source code is listed in the Appendix to this Detailed Description. Creation of an executable version of the SCROLL program involves the use of header files and utility routines whose source code is not listed in the appendices because they are unrelated to the present invention; one skilled in the art would appreciate the nature of and be capable of providing such supporting software.

Fig. 3 is a block diagram illustrating a logical arrangement of the scroll system. The components of this diagram are described in greater detail following a discussion of operational aspects of the system.

1 Operation

The scroll system presents to a user of the system a visually compressed display of a document. Fig. 4 is a picture of a display screen with a document displayed mostly in compressed form, with a window of legible text. The document that is displayed in Fig. 4 and in other similar figures is the computer program that is listed in the Appendix to this Detailed Description.

The display presented by the scroll system is somewhat like a vertically compressed display of a document over which are placed two bars that magnify vertically (as shown in Fig. 5), creating two windows of legible text. The user is able to discern gross shapes in the compressed text, and can read the text in each of the two magnified areas. Controls provided by the system permit the user to move the "magnifiers", and thereby select the portions of the document to be

readable. When the two windows are moved adjacent to each other, they become latched (as shown in Fig. 4) and can be moved as a single large window.

The greeked text is bordered (left and right) by white margins. So that a user can clearly see the location and extent of the windows, the windows are visually indicated in two ways: these windows extend horizontally beyond the margins of the greeking; there are short lines (through the margin of the greeking) on each side at the top and bottom of each window.

A user can control the visibility of tags, which can be displayed in legible form throughout the display. For example, Fig. 5 shows tags displayed as a result of a user's search for "`yofline`"; Fig. 6 shows a portion of the display of Fig. 5 at an expanded scale. Fig. 7 shows the display of two sets of tags that are embedded in the document. The system recognizes a word following the three-character sequence asterisk-digit-asterisk as a tag. For example, near the middle of Fig. 7 it can be seen that "`*1*`" identifies "`reopen`" as an embedded tag.

The figures illustrate the control that the user has over tag display. In Fig. 4 no tags are displayed. In Fig. 5 only tags showing results of a single search are displayed. In Fig. 7 embedded tags corresponding to digit 1 (e.g., "`reopen`") and to digit 2 (e.g., "`WINDOW`") are displayed. In the particular document displayed in these figures, "1" tags have been used to mark subroutine entry points and "2" tags have been used to mark major headings.

1.1 Greeking

The major portion of the display is text in greeked form, which is, in effect, a form of the original text compressed vertically by 16 times, with no horizontal compression. More particularly, while in the windows the text is displayed with a character size of 16 pixels (vertically) by 8 pixels (horizontally), each line of text corresponds to only one row of pixels in the greeking. Thus, the greeked portion of the display represents text at a 16-to-1 compression ratio.

Two characters of text correspond to a single unit of greeking, which is displayed as a single row of 16 pixels; a unit of greeking is the same width as that of the two characters being represented. If one or both of the two characters corresponding to a unit of greeking is/are a visible character (e.g., an alphanumeric or punctuation character), then that unit of greeking is displayed as alternating off

and on pixels (i.e., eight white pixels and eight black pixels in total); otherwise that unit of greeking is displayed as all white pixels.

The black pixels on odd lines are horizontally offset one pixel from the black pixels on even lines. This results in a checkerboard pattern that is perceived as gray. This has the advantage that when a tag is displayed in reverse video (white characters on a black background, which is reverse with respect to the text in the windows), the tag appears as a black rectangle against a gray background (in the greeking) or against a white background (in the windows).

Alternatively, greeking could be computed on the basis of a one-to-one correspondence between characters and units of greeking. However, when greeking is computed as described in the previous paragraph, single character spaces do not appear in the greeking, resulting in more visual emphasis on the overall shape of the text, without the distraction of holes created by individual spaces. This characteristic has been found to be useful.

There is another advantage to choosing the unit of greeking to be 16 pixels wide. If each unit of greeking is represented by a single pixel, the resulting display is compressed both horizontally and vertically by the same factor of 16.

Fig. 8 shows a portion of the display of Fig. 7 at a scale that makes individual pixels visible. Fig. 9, Fig. 10, Fig. 11, and Fig. 12 each show exactly the same portion of the display as shown in Fig. 8, but in each case the window of legible text has been moved down one line from the preceding figure. By comparison of succeeding figures in this sequence one can see the correspondence between visible characters and the greeking.

For example, by comparing Fig. 11 and Fig. 12, one can identify the unit of greeking in Fig. 12 that corresponds to the brace ("{") that appears on the left end of the window in Fig. 11. There are a series of isolated greeking units visible on the left side of these figures that each corresponds to a brace; these are each 16 pixels long (of which 8 are black), the width of two characters.

By comparing Fig. 8 and Fig. 9, one can see that the isolated space in "`UCHAR textbuf`" (see Fig. 9) is not manifest in the corresponding greeking (see Fig. 8). On the other hand, the three spaces between ">" and "`bo`" (in Fig. 12) does manifest itself in a single unit of white greeking (under "`xe. exc`" in Fig. 11).

1.2 Tags

Both in greeked text and in the windows, each tag appears at the horizontal location at which it occurs on its line of text. When a tag appears in the greeked text, it is approximately centered (vertically) on the greeked line to which it relates.

Tags may be identified by codes located in the text. Although the visibility of these tags may be turned off and on, these are "permanent" in the sense that they are embedded in the document itself. Other tags may be temporary; for example, tags can be created during the course of viewing the document by directing the system to search for particular text strings.

SCROLL provides for eight sets of tags. Fig. 7 shows the display of two sets of embedded tags: some are marked by "*1*" in the document and the other are marked by "*2*". Fig. 5 shows a third set of tags enabled, while the display of each of the two sets of embedded tags has been disabled; this third set is the result of a search for the string "yofline".

The figures Fig. 8 through Fig. 12 show how a tag moves as it transitions between a window and greeked text. From Fig. 8 to Fig. 9, the "reopen" tag moves up one text line. From Fig. 9 to Fig. 10, the tag makes a jump of half the height of visible text and moves out of the window. At this point (Fig. 10), the tag is associated with and centered on the first line of greeking above the window, but extends into the window. In Fig. 11 and Fig. 12, the window continues to move downward, while the tag remains centered on its line of greeking.

1.3 Window Control

The present invention is well suited to control by a pointing device; the illustrative embodiment of the invention uses a digitizing tablet and stylus. When the user touches the stylus to the tablet, the closest window moves (vertically) to the position on the display that corresponds to the position touched on the tablet. If the position that the window needs to move is relatively small (e.g., less than or equal to 100 lines), the window scrolls (i.e., the line-by-line movement is displayed; if the distance from the window's position to its target position is large (e.g., greater than 100 lines), it jumps to the target location without being displayed at any of the intermediate positions. Alternatively, window jumps could include display of a subset of the intermediate locations.

If the user holds the stylus down on the tablet and drags, the system attempts to make the closest window move to and track the stylus location. If the target location indicated by the stylus is too far from the window (e.g., more than 100 lines), then the system waits for the user to lift the stylus, at which point the window jumps to the target location.

If the present invention is employed with a tablet that overlays the display, the user can point (with the stylus) directly to the portion of the document that the user desires to see displayed in a window, and the system will move a window to that location. Further, the user can place the stylus "on" the window and drag it through the document.

When one of the windows becomes adjacent to the other window, the two windows become latched together: the marks between the two windows showing the vertical extent of the windows are eliminated, making the pair of windows appear visually as a single window that is twice the size of the individual windows; this double-size window then can be dragged as a single window. If the stylus touches down in the upper quarter of the combined window and drags upward, the window splits and the upper window follows the stylus; similarly, if the stylus touches down in the lower quarter of the window and drags downward, the window splits and the lower window follows the stylus.

1.4 Other Operational Aspects

A user starts SCROLL in the same manner as other programs are started when using MSDOS. In starting SCROLL, the user also specifies a parameter, which is the name of the file containing the document to be displayed.

When using SCROLL to browse through a document, the stylus is the primary control device. However, a keyboard is useful in such operations as requesting a search of the document for occurrences of a specified text string. To perform such an operation, the user presses the S key and the system prompts the user to type the text string for which a search is desired. After the user enters the string, the system searches the document for occurrences of the string, creates tags for each location in the document where the string is found, and then enables display of those tags, thereby making the search results visible.

Other keys provide control of the visibility of the various sets of tags. Each of the eight sets of tags provided by SCROLL can be individually enabled/disabled for

display. This is done with digit keys 1-8, each of which toggles the display state of a corresponding set of tags.

Although a pointing device is a convenient way to move the windows, there are occasions when keyboard control is useful. Although SCROLL does not provide for keyboard control of window position, the addition of keyboard control of window position is straightforward.

Pressing the q key causes SCROLL to terminate and return control to MSDOS.

2 Implementation

References are made in the following discussion to particular C functions, the source code of which can be found in the Appendix.

Many of the blocks in the diagram of Fig. 3 involve the execution of various portions of SCROLL. Some of the primary C functions involved in these blocks are as follows: `main()` in overall control; `reopen()`, `topup()`, `topdown()`, etc. in drawing control; `showgreek()` in greeking display; `typeset()` in legible display; `showtags()` in tag display.

2.1 Data Structures

During its operation, SCROLL uses a temporary file (used for the duration of execution of SCROLL) that contains the text of the input file arranged as an array of 80 character lines. The organization of this temporary file facilitates the many line-oriented operations that SCROLL performs.

Tag information is stored in a combination of several data structures. There is an array of tag entries, each of which stores the line and character position associated with a tag, an index into a list of ASCII character strings, and other information such as tag type. There is an array that functions as an index to the tag entries: this index being sorted according to the line on which a tag appears. There is a list of the character strings used to generate the displayed form of the tags.

The greeked form is stored in an array of 6-byte entries, one entry for each line to be displayed. The greeked form is stored as one bit per greeking unit. This is interpreted by `showgreek()`, which displays a row of 16 pixels for each bit representing a greeking unit: if the bit is set then one of two patterns of

alternating black and white pixels is displayed (the particular pattern depending upon whether the line is even-numbered or odd-numbered); otherwise the 16 pixels are all white.

Each 6-byte entry in the greeking array has 48 bits, which provides for representation of 40 greeking units for the 80 characters of a line and 4 units of margin on each side of the display. At initialization, `greekit()` sets the inner units of these margins to be "off" (white); when this array is interpreted by `showgreek()`, the outer 3 units are forced to solid black (not the half black that is used for greeking), resulting in a white border that is the width of a single unit of greeking (two character widths).

2.2 Flow of Control

The overall flow of the SCROLL program is illustrated in Fig. 13. The operation of window movement according to stylus position is further detailed in Fig. 15.

When SCROLL is started, it performs certain initialization operations and then enters an interactive loop that continues until the user "quits" by use of the q key (see `main()`).

The initialization includes reading the input file and creating the temporary file that is organized by 80 character lines. This part of the initialization is managed by `maketemp()`. As each line is added to the temporary file, the line is also scanned for tags (see `findtags()`) and the greeked form is created (see `greekit()`).

The last major step of initialization is the drawing of the initial display of greeked text with the text windows in their initial positions; this is done by calling `reopen()`. In the illustrative embodiment, the windows always begin at the same two locations. The system could be arranged to set the initial window positions according to some information pertaining to the user's access to the document — e.g., the first match of a search, the locations of the windows the last time the document was displayed.

The interactive loop is repeated until a variable "`quitting`" is set.

The interactive loop begins by testing for a keystroke from the user (see Fig. 14).

If a key has been pressed that corresponds to a set of tags that are currently displayed, then the display state for that set of tags is toggled to "off" and the display is redrawn. If, instead, a key has been pressed that corresponds to a set of tags not currently displayed, then the display state for that set of tags is toggled "on" and only the tags are redrawn.

If the q key has been pressed then "`quitting`" is set, indicating that this will be the last time through the interactive loop.

If the user has requested a search (the s key), then the user is prompted to enter the string to be searched, after which the temporary file is searched, the tag data structures updated appropriately, and the tags are redrawn. The system of the illustrative embodiment includes a second display screen, on which the prompt and response are displayed. Alternatively, this could be done on the same screen on which the document is displayed, either in a portion of the screen reserved for user interaction or directly over the document display; in the later case, at least the overwritten portion of the document display must be redrawn.

After keystroke processing, stylus processing and window movement is performed (see Fig. 15).

If the stylus is down, the next step in the interactive loop is to consider the stylus position. If the stylus has just touched down since the last time through the loop, then the window closest to the line to which the stylus is pointing becomes the moving window. The targeted center position for the currently moving window is set to the line of legible text or greeking to which the stylus is pointing.

The last step in the interactive loop depends on whether either of the windows is too far from its target location. If one of the windows is farther than 100 lines from its target position, the system waits for the user to lift the stylus from the tablet, after which the display is redrawn with the windows at their target locations. Otherwise, the currently moving window is scrolled by one line toward its target position.

When a window is scrolled, the display is updated as follows: the text window is drawn, the greeking near the window is redrawn (because tags could have jumped out of this greeking), and the tags are redrawn in a range around and including the window are redrawn. Because redrawing of the nearby greeking can overwrite

a tag, the range of redrawn tags extends beyond those that overlap the window; it includes tags that overlap the redrawn greeking.

If the user did not press the q key, the interactive loop begins again.

When the interactive loop is terminated, after some cleanup, the SCROLL program terminates its execution.

3 Second Level of Visual Compression

The scroll system uses a display of 1024 lines and, as described above, provides for display of approximately 13 pages of a document. A variation on the system described above can display much larger documents (or larger portions of yet larger documents) by using a second level of visual compression. The greeking described above effects visual compression by a factor of 16 in a single dimension.

By compressing horizontally, as well as vertically, more text can be displayed on the same screen. For example, Fig. 16 illustrates a screen display that has text at each of two different levels of visual compression. The small chains of boxes at the top, middle, and bottom of the figure represent portions of text that is visually compressed horizontally and vertically. These boxes could display the type of greeking described above, but with only one pixel per greeking element. The remaining portions of the display includes windows of legible text movable within regions of text that are compressed in only the vertical dimension, as described above. As illustrated in Fig. 16, the amount of text between the windows may be such that the second level of compression is used for some of the text between the windows. When the windows are much closer, the second level of compression would disappear from the middle of the document. Similarly, depending upon window position, the boxes representing text at the second level of visual compression may disappear at either end of the display. The number of lines in each of the three rows of boxes is adjusted depending upon the positions within the document of the windows of legible text. When a window comes close to the edge of its region of one-dimensional greeking, the display is readjusted, generally lengthening the small boxes in one of the three rows and shortening the small boxes in another of the three rows. In addition, if the user touches a place corresponding to one of the little boxes, the display is rearranged, so as to move a window to that part of the document to which the user has pointed.

The locations of tags, as described above, can provide a user with valuable information (e.g., context that aids the user in identifying locations within the document, or information about how the matter identified by the tags is distributed through the document). These tags can be displayed in the small pages, as well as in the portions of the document in and near the windows.

If the type of greeking described above is also used for the small "pages", then these pages provide about an order of magnitude more compression than in the portion of the display that is only compressed in the vertical dimension. (Within one of the page boxes, the increase in compression is a factor of 16; however because of the space between the boxes the effective compression is less.)

Even larger documents could be displayed by using a more aggressive compression scheme for the second level of compression. For example, a single line of greeking can represent more than one line of text.

4 Additional Variations

The implementation described above illustrates the use of the inventive scrolling mechanisms outside the context of any particular application. However, these mechanisms are well suited for incorporation into systems for performing specific functions, such as the following examples. The present display arrangement can be used for a document editor; the present inventive display capabilities can be added to an editor, or editing capabilities can be added to the illustrative embodiment described above. It can be used by a window manager, where the window manager is responsible for providing the user ways to view (through windows) "screens" larger than the window through which they are being viewed. The present invention could be used with a data retrieval system when search results are too extensive to be legible on the screen at one time; the tags could be used to identify locations of words that were the subject of the search, showing the words as they actually appear at their respective locations in the search results (for example, in the case where the retrieval system uses a thesaurus to search for words in addition to those supplied by the user, the word actually found in the document would be displayed).

The illustrative embodiment described above displayed tags that were visible throughout. Similarly, other mechanisms can be used for indicating information in the document. For example, color could be used; like the tags, this color could

appear both in the legible portion of the document and in the greeked portion of the document. Black (rather than gray) greeking could be used to highlight portions of the document. The color or other highlighting could be of various significance, e.g., text marked by the user, type of information (e.g., data structures, comments, subroutines), regions of redaction, authorship, vintage.

Further types of tags would be useful in certain applications. For example, it may be useful to provide for markers that a user can insert to remember places of interest in a document. In documents that have meaningful page boundaries, tags could be included that mark these pages, permitting the user to go directly to a particular page by placing the stylus at the corresponding page tag. These page tags could appear along the edge of the document display. In shared documents, sets of tags may be personal to individual users.

Provision could also be made for a user to add handwritten annotations to the document. In this case either the system would save all or annotated portions of the document in bitmap form (rather than ASCII), or would save the annotations as vectors.

The focus of the detailed discussion above has been on a system using two expansion windows. However, the present invention can be employed with only a single such window and can be employed with more than two windows.

As indicated above, the invention can be implemented on computer systems other than that described in the illustrative embodiment above. In particular, one can implement this using the 640 pixel by 480 pixel mode of a VGA-type display. In that case, it is appropriate to use smaller characters (e.g., 8x6 pixel character cells) so that a display can be created that is comparable in scope to that achieved on the 1024x1024 display of the illustrative embodiment.

Although the illustrative embodiment utilizes a stylus and digitizing tablet, the invention can be implemented with other pointing devices. For example, a mouse can be used. With reduced utility, even a keyboard can be used.

The term "document" has been used to describe the body of data through which the present invention provides navigation. In the present context, "document" includes any body of data that includes text, such as text files, word processing documents, database records, compound documents. The data of a "document" may be stored in a file on a mass storage device, or may be more transitory in

nature (e.g., database search results assembled at the time the user requests their presentation).

In the illustrative embodiment, a line of the greeked representation corresponds to a line of characters in the document; further, a line of the greeked representation is a single row of pixels. Alternatively, two or more lines of characters can be compressed into a single line of the greeked representation. Further, a line of the greeked representation can be displayed as two or more rows of pixels. In addition, one skilled in the art will appreciate that other methods for computing the greeking could be used to emphasize or de-emphasize particular aspects of documents.

Although the invention has been described in the context of a specific embodiment, additional variations and alternative implementations will be apparent to those skilled in the art. Thus, the invention is not limited to the specific details and illustrative examples shown and described in this specification. Rather, it is the object of the appended claims to cover all such variations and modifications as come within the true spirit and scope of the invention.

APPENDIX

```
/* scroldoc.c 3/6/91 document scroller                          ken knowlton
 *                                          (c) Copyright 1991 Wang Labs Inc.
 *
 *      ----------------INITIALIZATION--------------------
 *      main(argc,argv)                 Usage: >scroldoc infile
 *      int  argc;
 *      char    **argv;
 *
 *      maketemp(infd,tempfd)           makes temp file, expand tabs. Lines out
 *      int      infd,tempfd;           are 80 chars. makes greeking & tag list
 *
 *      UCHAR *expline(from, to)        expands line, up to 80 chars to 'to'.
 *      UCHAR         *from,*to;        returns updated 'from' after next EOL.
 *
 *      gettext(tempfd,line,buf)        retrieves 80-character textline from
 *      int     tempfd;                 tab-expanded, 80-wide ascii file whose
 *      short          line;            file descriptor is tempfd
 *      UCHAR              *buf;
 *
 *      ----------------WINDOW MOVE-----------------------
 *      reopen(topcent,botcent)         re-create screen with windows centered
 *      short  topcent,botcent;         on specified textlines
 *
 *      topup()                         top window up one line
 *
 *      topdown()                       top window down one (assumes room)
 *
 *      botup()                         bottom window up one (assumes room)
 *
 *      botdown()                       bottom window down one
 *
 *      combup()                        combined (joined) window up one
 *
 *      combdown()                      combinad (joined) window down one
 *
 *      ----------------GREEKING--------------------------
 *      greekit(from, to)               greek the 80-char line, represented as
 *      USHORT *from;                   2 chars/word, padded to 48 wds. Each wd
 *      UCHAR        *to;               compressed to a bit (1 line to 6 bytes)
 *
 *      showgreek(from,thru)            displays greeked lines of stated range
 *      short     from,thru;            that are NOT part of either window
 *
 *      ----------------TAG MANAGEMENT--------------------
 *      findtags(line)                  searches 80-char line for sequence *n*
 *      UCHAR   *line;                  and adds tags found to TAG tags[].
 *
 *      addtags(string)                 re-search of infile for instances of
 *      UCHAR  *string                  string, adds those found to tag list
 *
 *      insertag(xl,li,st,token)        inserts tag "token" at end of tags[]
 *      short    xl,li,st;              with specified xloc, line, and state;
 *      UCHAR             *token;       tagdexes[] entry bubbles to line order
 *
 *      showtags(from,thru,estimatedex) refreshes tags on lines 'from' thru
 *      short    from,thru,estimatedex; 'thru', start looking at estimatedex.
 *
 *      puttag(tagindex)                put on screen tag whose index is given
 *      short  tagindex;
```

```
*
*       ----------------MAPPING: TEXTLINE<-->SCREEN--------
*       yofline(textline)               says where on screen center of a text
*       short   textline;               line is (function of window positions)
*
*       lineofy(screeny)                identifies text line that stylus at
*       short   screeny;                screen y is pointing to
*
*       ----------------MISC. EXTERNAL UTILITIES-----------
*       grafstart()                     switch screen: text to graphics mode
*
*       grafstop()                      switch screen: graphics to text mode
*
*       pendown()                       returns 1 iff stylus down on x-y tablet
*
*       peny()                          returns y value of stylus on x-y tablet
*
*       solidrect(lef,top,rit,bot,col)  make a solid filled rectangle, black if
*       short     lef,top,rit,bot,col;  col==Black, else white if col==White.
*
*       typeset(ascii, to,delta,bonw)   puts 8x15 patterns for ascii string to
*       UCHAR  *ascii,*to;              buffer of width 'delta' bytes, iff bonw
*       short             delta,bonw;   black=1 on white=0, else white on black
*
*       outblt(buf,wide,high,x,y)       blt to screen.  Source wide,high in
*       UCHAR *buf;                     pixels;  dest x,y is top left in abso-
*       short      wide,high,x,y;       lute screen coordinates
*/

#include   <stdio.h>    /* C: standard I/O      */
#include   <fcntl.h>    /* C: file control      */
#include      <io.h>    /* C: I/O               */

/* manifest constants & macros */
#define  UCHAR   unsigned char
#define  USHORT  unsigned short
#define MAXLINES          860
#define MAXTAGS           500
#define WORDSPACE        4000
#define GREEKREDO          12
#define TAGREDO            21
#define BLACKONWHITE        1
#define WHITEONBLACK        0
#define S_IREAD       0000400       /* file permission: read        */
#define S_IWRITE      0000200       /* file permission: write       */
#define Black               1
#define White               0

#define alpha(x)       ( ((x) >= 'a' && (x) <= 'z') ||                  \
                         ((x) >= 'A' && (x) <= 'Z') ||  (x) == '_' )
#define inwindow(line)   ( ((line) > topmidlin-3 && (line) < topmidlin+3) || \
                           ((line) > botmidlin-3 && (line) < botmidlin+3) )

/* prototypes for routines returning non-short values */
UCHAR   *expline(UCHAR *,UCHAR *);

short toptarg = 50, bottarg = 55;       /* targeted topmidlin,botmidlin */

/* window buffer array & globals. */
union{
     UCHAR  byts[16*10][80];    /* text window(s): one of 10 lines or   */
     USHORT wrds[16*10][40];    /*                 two of  5 lines each */
  }bltwind;
short topmidlin = 20, botmidlin = 25; /* textline of top,bot window ctrs*/
short joined    =   1;          /* are the windows joined?              */
short maxline   = 860;          /* maximum lines we can deal with       */
short movetop   =   1;          /* is the moving window the top one?    */
```

```
short stilldown =   0;          /* is pen still down?                    */

/* infile & outfile data, No. of lines, greeking */
int   infd,tempfd;              /* input & outfile descriptors           */
short inplines = 0;             /* number of input lines                 */
UCHAR greeking[MAXLINES][6];    /* compressed greeking 6 bytes per line*/

/* tags: array and other globals */
typedef struct{
        short xloc;             /* position in the line     0-79         */
        short line;             /* document line number     0-MAXLINES   */
        short where;            /* index into words for ascii string     */
        short state;            /* misc. state info: kind, visible...    */
   }TAG;
TAG       tags[MAXTAGS];        /* labels, free if .xloc == -1           */
short tagnum    = 0;            /* number of tags found so far           */
short nextokdex = 0;            /* place to store ascii of the next tag */
short tagdexes[MAXTAGS];        /* y-ordered indexes into labels[]       */
short whoshow[10] = { 0,0,0,0,0,0,0,0,0,0 }; /*which kinds of tags show?*/
short nextkind  = 3;            /* 1,2 embedded, 3,4..interactive search*/
char words[WORDSPACE+35];       /* list of tag words, each ends w. '\0' */

/* Estimates of where in tagdexes[] to find tags for current tag-update span */
short   toptagdex = 0;  /* index into tagdexes[] of first tag for top window */
short   bottagdex = 0;  /* index into tagdexes[] of first tag for bot window */


                /*2*/ /* MAIN PROGRAM AND INITIALIZATION */
/*1*/ main(argc,argv)   /* Note: "comments" 2 & 1 tag 'MAIN' & 'main' resp. */
int         argc;
char          **argv;
{
        short i,j,quitting,targline,topdist,botdist;
        UCHAR inch,newtag[25],bltbuf[6];

        /* usage check */
        if(argc != 2){  printf("\n Usage:  >scroldoc  infile"); exit(1); }

        /* open infile */
        infd = open(argv[1],O_RDONLY | O_BINARY);
        if(infd <= 0){  printf("\n can't open infile %s",argv[1]); exit(1); }

        /* open outfile, process infile to outfile */
        tempfd = open("temp",O_TRUNC|O_BINARY|O_RDWR|O_CREAT,S_IREAD|S_IWRITE);
        if(tempfd <= 0){ printf("\n can't open file 'temp'"); exit(1); }
        maketemp(infd,tempfd);

        /* initialize tablet+graphics system */
        grafstart();

        /* interactive loop */
        reopen(50,55);          /* starting positions for windows */
        quitting = 0;
        while(!quitting){

                inch = key();   /* keyboard control */
                switch(inch){
                    case '1':  case '2':  case '3':  case '4':  /* toggle   */
                    case '5':  case '6':  case '7':  case '8':  /* tags 1-8 */
                        i = inch - '0';                 /* which is it?     */
                        whoshow[i] = !whoshow[i];       /* toggle its state */
                        if(!whoshow[i])reopen(topmidlin,botmidlin);
                        else           showtags(0,inplines-1,0);
                        break;
                    case 'q':   /* stop */
                        quitting = 1;
                        break;
```

```
            case 's':   /* search and enter new tags in list */
                if(nextkind > 8)break;          /* too many kinds */
                printf("\n enter string to search for: ");
                scanf("%s",newtag);
                printf("\n looking for '%s'",newtag);
                addtags(newtag);
                showtags(0,inplines-1,0);
                break;
        }

        /* reinterpret which window is moving & its targeted y level */
        if(pendown()){  /* if pen down ... */
                targline = lineofy(peny());

                /* if newly down, re-ID moving window*/
                if(!stilldown){
                        topdist = targline - topmidlin;
                        botdist = targline - botmidlin;
                        if(topdist < 0)topdist = -topdist;
                        if(botdist < 0)botdist = -botdist;
                        movetop = topdist < botdist; /* which moves? */
                        stilldown = 1;

                        /* if pen down so as to separate, unjoin. */
                        if(targline < topmidlin ||
                           targline > botmidlin)joined = 0;

                }
                if(movetop)toptarg = targline;
                else       bottarg = targline;
        }
        else stilldown = 0;     /* else pen not down */

        /* if either has far to go, wait for penup, reopen, stabilize*/
        if(topmidlin-toptarg > 100 || toptarg-topmidlin > 100 ||
           botmidlin-bottarg > 100 || bottarg-botmidlin > 100){
                while(pendown()){
                        if(movetop)toptarg = lineofy(peny());
                        else       bottarg = lineofy(peny());
                }
                if(movetop && joined)reopen(toptarg-3,toptarg+2);
                else if(      joined)reopen(bottarg-3,bottarg+2);
                else                 reopen(toptarg  ,bottarg  );
                toptarg = topmidlin;  /* declare both windows happy */
                bottarg = botmidlin;  /*      to be where they are   */
        }

        /* else move the moving window(s) toward targeted position */
        else{
                if(movetop){
                        if     (toptarg < topmidlin)topup();
                        else if(toptarg > topmidlin)topdown();
                        bottarg = botmidlin;    /* declare bot happy */

                }
                else{
                        if     (bottarg < botmidlin)botup();
                        else if(bottarg > botmidlin)botdown();
                        toptarg = topmidlin;    /* declare top happy */
                }
        }
    }
    /* close up shop */
    close(infd);            /* original text file                    */
    close(tempfd);          /* 80-line-wide ascii version of file    */
    grafstop();             /* graphics system: revert to text mode */
}

/* make 'temp' file (infd & tempfd freshly opened). Uses bltwind.byts[][] */
```

```
/*1*/maketemp(infd,tempfd)
int           infd,tempfd;
{
        union{  UCHAR byts[96];  USHORT wrds[48]; }expanded;
        UCHAR *inbuf,*begptr;   /* inbuf ptrs: begin, next line,        */
        UCHAR *endptr;          /*                          current end  */
        short extra,didread,i;

        inbuf = bltwind.byts[0];
        memset(inbuf,'\0', 600);
        expanded.byts[88] =  'a';  /* guarantee stop for *n* string     */
        expanded.byts[89] = '\0';  /* guarantee other scanning stops    */
        expanded.byts[90] = '\0';
        begptr = endptr = inbuf;
        extra   = 0;
        didread = 1;    /* fake prev read: at start input not exhausted */

        /* loop while did read some last time or still have some in buffer */
        while(extra>0 || didread>0){

                /* if < 80 chars remain, try to get up to 512 more input */
                if(extra<80 && didread>0){
                        if(begptr != inbuf){
                                if(extra)memmove(inbuf,begptr,extra);
                                begptr  = inbuf;
                                endptr  = begptr + extra;
                        }
                        didread =  read(infd,endptr,512);
                        endptr  += didread;
                        *endptr     = *(endptr+2) = 0x0d;  /* guarantee that */
                        *(endptr+1) = *(endptr+3) = 0x0a;  /* scanning stops */
                }
                /* process: expand, write, find tags, greek, inplines++ */
                begptr = expline(begptr,&expanded.byts[8]);
                write(tempfd,&expanded.byts[8],80); /*write out expanded line*/
                findtags(&expanded.byts[8]);
                greekit(expanded.wrds,greeking[inplines]);

                /* prepare for next line */
                inplines++;
                if(inplines >= MAXLINES)break;
                if(begptr > endptr)break;
                extra = endptr - begptr;
        }
        return;
}

/* expand line, up to 80 chars, returns updated 'from' after next EOL */
UCHAR /*1*/ *expline(from, to)
UCHAR               *from,*to;
{
        short putdex,thisone;

        memset(to,' ',80);      /* preset 80-character line to blanks */
        putdex = 0;
        while(putdex<80){
                thisone = *from++;
                if      (thisone == 0x0d)break;
                else if(thisone == '\t'){    /* expand tab: at least 1 blank */
                        while(++putdex%8);
                }
                else *(to + putdex++) = thisone;
        }

        /* scan forward over next 0d 0a */
        while(*from != 0x0a)from++;
        return(++from);
```

```
}

/* get 80-char line of ascii text */
gettext(fd,line,buf)
int     fd;
short      line;
UCHAR           *buf;
{
        lseek(fd, (long)line*80, 0);
        read(fd,buf,80);
        return;
}

                /*2*/ /* WINDOW MOVE */
/* reopen windows */
/*1*/ reopen(topcen,botcen)
short       topcen,botcen;
{
        short i,temp,tline,yout,dy,lastgreek;
        UCHAR textbuf[81];

        /* legalize. exchange identities if crossed over. limit at file size */
        if(topcen >   botcen){ temp = topcen; topcen = botcen; botcen = temp; }
        if(topcen <        2)topcen =              2;     /* highest allowed   */
        if(botcen>inplines-9)botcen = inplines - 9;     /* lowest allowed    */
        if(botcen < topcen+5)botcen = topcen   + 5;     /* min separation    */

        /* update window globals */
        if(botcen <= topcen+5)joined = 1;  /* join windows if 5 apart */
        topmidlin = topcen;
        botmidlin = botcen;
        toptagdex = bottagdex = 0;

        /* clear window buffer, typeset with 10 text lines */
        memset(bltwind.byts[0], 0, 80*10*16);   /* clear to zero */
        textbuf[80] = '\0';
        for(tline=0; tline<5; tline++){
                gettext(tempfd,tline+topmidlin-2,textbuf);
                typeset(textbuf,bltwind.byts[16*tline+1],80,BLACKONWHITE);
        }
        for(tline=0; tline<5; tline++){
                gettext(tempfd,tline+botmidlin-2,textbuf);
                typeset(textbuf,bltwind.byts[16*(tline+5)+1],80,BLACKONWHITE);
        }
        solidrect(0,0,1023,1023,Black); /* initialize screen: black        */

        /* show top greeking if any */
        showgreek(0,topmidlin-3);

        /* if joined, show combined window */
        if(joined){
                yout = 1 + yofline(topmidlin-3);
                solidrect(192-32, yout, 192+639+32, yout+159, White);
                outblt(bltwind.byts[0],640,16*10,192,yout);
        }
        /* else show top window, then middle greeking, then bottom window */
        else{
                yout = 1 + yofline(topmidlin-3);
                solidrect(192-32, yout, 192+639+32, yout+79, White);
                outblt(bltwind.byts[ 0  ],640,16*5,192,yout);
                showgreek(topmidlin+3,botmidlin-3);
                yout = yofline(botmidlin-2) - 8;
                solidrect(192-32, yout, 192+639+32, yout+79, White);
                outblt(bltwind.byts[16*5],640,16*5,192,yout);
        }

        /* show bottom greeking if any, then show all tags */
```

```
        showgreek(botmidlin+3, inplines<850 ? inplines : 850);
        showtags(0,inplines-1,0);
        return;
}

/* move top window up one */
/*1*/ topup()
{
        UCHAR textline[81];
        short ytop,i;

        if(topmidlin <= 2)return;

        /* if windows joined, do by combup() */
        if(joined){     combup();       return; }

        /* move top bufferful down a line, get and typset the new line */
        topmidlin--;
        memmove(bltwind.byts[16],bltwind.byts[0],4*16*80);
        gettext(tempfd,topmidlin-2,textline);
        textline[80] = '\0';
        typeset(textline,bltwind.byts[1],80,BLACKONWHITE);

        /* blt top five lines to screen */
        ytop = 1 + yofline(topmidlin-3);
        solidrect(160, ytop, 863, ytop, White); /* widen */
        outblt(bltwind.byts[0],640,5*16,192,ytop);

        /* update nearby greeking and tags */
        showgreek(topmidlin-GREEKREDO,topmidlin+GREEKREDO);
        toptagdex = showtags(topmidlin-TAGREDO, topmidlin+TAGREDO, toptagdex);
        return;
}

/* move top window down one */
/*1*/ topdown()
{
        UCHAR textline[81];
        short ytop,i;

        if(topmidlin > maxline)return;

        /* if physically together, call them joined */
        if(topmidlin >= botmidlin - 5)joined = 1;

        /* if joined, move both down together by combdown() */
        if(joined){     combdown();     return; }

        /* move top buffer up a line, get and typset the new line */
        topmidlin++;
        memmove(bltwind.byts[0],bltwind.byts[16],4*16*80);
        gettext(tempfd,topmidlin+2,textline);
        textline[80] = '\0';
        typeset(textline, bltwind.byts[1+4*16], 80, BLACKONWHITE);

        /* blt top five lines to screen */
        ytop = 1 + yofline(topmidlin-3);
        solidrect(160, ytop+79, 863, ytop+79, White);   /* widen */
        outblt(bltwind.byts[0],640,5*16,192,ytop);

        /* windows may now be joined */
        if(topmidlin+5 >= botmidlin)joined = 1;

        /* update nearby greeking and tags */
        showgreek(topmidlin-GREEKREDO,topmidlin+GREEKREDO);
        toptagdex = showtags(topmidlin-TAGREDO, topmidlin+TAGREDO, toptagdex);
        return;
```

```
}

/* move bottom window up one */
/*1*/ botup()
{
        UCHAR textline[81];
        short ytop,i;

        /* if physically together, call them joined */
        if(botmidlin <= topmidlin + 5)joined = 1 ;

        /* if joined, move both up together by combup() */
        if(joined){     combup();       return; }

        /* move bot buffer down a line, get and typset the new line */
        botmidlin--;
        memmove(bltwind.byts[16*6],bltwind.byts[16*5],4*16*80);
        gettext(tempfd,botmidlin-2,textline);
        textline[80] = '\0';
        typeset(textline, bltwind.byts[1+5*16], 80, BLACKONWHITE);

        /* blt bottom five lines to screen */
        ytop = yofline(botmidlin-2) - 8;
        solidrect(160, ytop, 863, ytop, White); /* widen */
        outblt(bltwind.byts[16*5],640,5*16,192,ytop);

        /* windows may now be joined */
        if(topmidlin+5 >= botmidlin)joined = 1;

        /* update nearby greeking and tags */
        showgreek(botmidlin-GREEKREDO,botmidlin+GREEKREDO);
        bottagdex = showtags(botmidlin-TAGREDO, botmidlin+TAGREDO, bottagdex);
        return;
}

/* move bottom window down one */
/*1*/ botdown()
{
        UCHAR textline[81];
        short ytop,i;

        if(botmidlin+3 >= inplines)return;

        /* if joined, move both by combdown() */
        if(joined){     combdown();     return; }

        /* (else) move bot buffer down a line, get and typset the new line */
        botmidlin++;
        memmove(bltwind.byts[16*5],bltwind.byts[16*6],4*16*80);
        gettext(tempfd,botmidlin+2,textline);
        textline[80] = '\0';
        typeset(textline, bltwind.byts[1+9*16], 80, BLACKONWHITE);

        /* blt bottom five lines to screen */
        ytop = 1 + yofline(botmidlin-3);
        solidrect(160, ytop+79, 863, ytop+79, White);   /* widen */
        outblt(bltwind.byts[16*5],640,5*16,192,ytop);

        /* update nearby greeking and tags */
        showgreek(botmidlin-GREEKREDO,botmidlin+GREEKREDO);
        bottagdex = showtags(botmidlin-TAGREDO, botmidlin+TAGREDO, bottagdex);
        return;
}

/* move combined windows up -- very much like topup() */
/*1*/ combup()
{
```

```
        UCHAR textline[81];
        short ytop,i;

        if(topmidlin <= 2)return;
        topmidlin--;
        botmidlin--;

        /* move entire buffer down a line, get and typset the new line */
        memmove(bltwind.byts[16],bltwind.byts[0],9*16*80);
        gettext(tempfd,topmidlin-2,textline);
        textline[80] = '\0';
        typeset(textline,bltwind.byts[1],80,BLACKONWHITE);

        /* blt entire ten lines to screen */
        ytop = 1 + yofline(topmidlin-3);
        solidrect(160, ytop, 863, ytop, White); /* widen */
        outblt(bltwind.byts[0],640,10*16,192,ytop);

        /* update/refresh nearby greeking and tags */
        showgreek(topmidlin-GREEKREDO, botmidlin+GREEKREDO);
        toptagdex = showtags(topmidlin-TAGREDO, botmidlin+TAGREDO, toptagdex);
        bottagdex = toptagdex;
        return;
}

/* move combined window down one -- very much like botdown() */
/*1*/ combdown()
{
        UCHAR textline[81];
        short ytop,i;

        if(topmidlin+3 >= inplines)return;
        topmidlin++;
        botmidlin++;

        /* move entire buffer up a line, get and typset the new line */
        memmove(bltwind.byts[0],bltwind.byts[16],9*16*80);
        gettext(tempfd,botmidlin+2,textline);
        textline[80] = '\0';
        typeset(textline,bltwind.byts[1+9*16],80,BLACKONWHITE);

        /* blt entire ten lines to screen */
        ytop = 1 + yofline(topmidlin-3);
        solidrect(160, ytop+159, 863, ytop+159, White); /* widen */
        outblt(bltwind.byts[0],640,10*16,192,ytop);

        /* update/refresh nearby greeking and tags */
        showgreek(topmidlin-GREEKREDO, botmidlin+GREEKREDO);
        toptagdex = showtags(topmidlin-TAGREDO, botmidlin+TAGREDO, toptagdex);
        bottagdex = toptagdex;
        return;
}

                /*2*/ /* GREEKING */
/* greek the 80 character line to 6 bytes, high order bit is picture left */
/*1*/ greekit(from, to)
USHORT      *from;
UCHAR           *to;
{
        short group,byt,bit;

        /* double space 2020 goes to zero bit, other pairs to 1 bits */
        for(group=0; group<6; group++){
                byt = 0;
                for(bit=0x01; bit < 0x100; bit <<= 1){
                        if(*from++ != 0x2020)byt |= bit;
                }
```

```
                if(group==0)byt = byt & 0xF3;           /* white borders     */
                if(group==5)byt = byt & 0xCF;           /*   right and left  */
                *to++ = byt;
        }
        return;
}

/* show greeking for the range, skipping lines in windows */
/*1*/ showgreek(from,thru)
short                   from,thru;
{
        USHORT *buf,putword,greekbuf[48];
        short getdex,getbyt,lin,bit;

        /* legality trim */
        if(from <          0)from = 0;
        if(thru >= inplines)thru = inplines - 1;
        if(thru < from)return;

        /* iterate through the lines, skipping windows */
        for(lin=from; lin <= thru; lin++){
                if(inwindow(lin))continue;
                buf = greekbuf;
                putword = lin%2 ? 0x5555 : 0xAAAA;
                for(getdex=0; getdex<6; getdex++){
                        getbyt = greeking[lin][getdex];
                        for(bit=0x01; bit < 0x100; bit <<= 1){
                                *buf++ = getbyt & bit ? putword : 0;
                        }
                }
                greekbuf[ 0] = greekbuf[ 1] = greekbuf[ 2] = 0xFFFF;
                greekbuf[45] = greekbuf[46] = greekbuf[47] = 0xFFFF;
                if(lin==topmidlin-3 || lin==topmidlin+3 ||
                   lin==botmidlin-3 || lin==botmidlin+3){
                        greekbuf[3] = greekbuf[44] = 0xFFFF;
                }
                outblt((UCHAR *)greekbuf,640+128,1,192-64,yofline(lin));
        }
        return;
}

                /*2*/ /* TAG MANAGEMENT */
/* search 80-char line and add any strings preceeded by *n* to taglist */
/*1*/ findtags(line)
UCHAR           *line;
{
        short getdex,putdex,thisone,digit,column,nexnex,scandex;
        UCHAR token[35];

        if(tagnum >= MAXTAGS || nextokdex >= WORDSPACE)return;
        for(getdex=0; getdex<77; getdex++){
                thisone = *(line + getdex);
                if(thisone != '\*')continue;
                digit   = *(line + getdex + 1);
                if(digit < '0' || digit > '9')continue;
                nexnex  = *(line + getdex + 2);
                if(nexnex != '\*')continue;

                /* yes, this is first * of *n*, scan over non-alpha */
                scandex = getdex + 3;
                do{
                        thisone = *(line + scandex++);
                }while(!alpha(thisone));

                /* collect. lower = firstchar, scandex on next */
                column = scandex - 1;
                putdex = 0;
```

```
                token[putdex++] = thisone;
                while(thisone = *(line+scandex++),alpha(thisone) && putdex<34){
                        token[putdex++] = thisone;
                }
                token[putdex] = '\0';

                /* make the entry in word list */
                insertag(column,inplines,digit-'0',token);
        }
        return;
}

/* search of temp file to find instances of string; adds tags to list */
/*1*/ addtags(string)
UCHAR        *string;
{
        short lin,length,i,dx,toofar;
        UCHAR ch,starter,original,token[200],inline[82];

        /* count string length, set chars to upper */
        length = 0;
        while(string[length] != '\0')length++;          /* count        */
        for(i=0; i<length; i++)string[i] &= 0xDF;       /* to upper     */
        starter = string[0];                            /* 1st char     */

        /* treat line by line */
        inline[0] = inline[81] = '\t';  /* stop extension left & right */
        lseek(tempfd, (long)0, 0);
        toofar  = 81 - length;
        for(lin=0; lin<inplines; lin++){
                read(tempfd,&inline[1],80);

                /* scan the line looking for possible starting points */
                for(i=1; i<toofar; i++){
                        if((inline[i] & 0xDF) != starter)continue;

                        /* start collecting (possible) token at index 100 */
                        dx = 0;
                        while(((original=inline[i+dx]) & 0xDF) == string[dx]){
                                if(string[dx] == '\0')break; /*' '&0xDF=='\0'*/
                                token[100 + dx++] = original; /* orig case */
                        }
                        /* fail if string not finished, else success */
                        if(string[dx] != '\0')continue;

                        /* found, extend in both directions */
                        while(alpha(original)){
                                token[100+dx++] = original;
                                if(dx>95)break;
                                original = inline[i+dx];
                        }
                        token[100+dx] = '\0';   /* terminate token */
                        dx = -1;
                        original = inline[i+dx];
                        while(alpha(original)){
                                token[100+dx--] = original;
                                if(dx < -95)break;
                                original = inline[i+dx];
                        }
                        /* token starts at token[i+dx], x = one less */
                        insertag(i+dx,lin,nextkind,&token[100+dx+1]);
                }
        }
        whoshow[nextkind] = 1;  /* make them visible */
        nextkind++;
        return;
}
```

```
/* insert tag to end of tags[], pointer to end tagdexes[] bubbles back */
/*1*/ insertag(xl,li,st,token)
short           xl,li,st;                   /* xloc, line, state    */
UCHAR                   *token;             /* pointer to string    */
{
        short bubdex,temp;

        if(tagnum >= MAXTAGS || nextokdex >= WORDSPACE)return;
        tags[tagnum].xloc  = xl;
        tags[tagnum].line  = li;
        tags[tagnum].where = nextokdex;
        tags[tagnum].state = st;
        strcpy(&words[nextokdex++],token);
        while(words[nextokdex++] != '\0');
        bubdex = tagdexes[tagnum] = tagnum;

        /* bubble up: run backwards interchanging til in order (or at top) */
        while(bubdex>0 && tags[tagdexes[bubdex  ]].line <
                          tags[tagdexes[bubdex-1]].line   ){
                temp                = tagdexes[bubdex-1];
                tagdexes[bubdex-1] = tagdexes[bubdex  ];
                tagdexes[bubdex  ] = temp;
                bubdex--;
        }
        tagnum++;
        return;
}

/* show tags for lines 'from' thru 'thru'; est = estimatedindex */
/*1*/ showtags(from,thru,est)
short           from,thru,est;
{
        short startdex,tagindex;

        /* legalize: point to an actual entry (if none, return) */
        if(tagnum <=       0)return;
        if(est    <        0)est = 0;
        if(est    >= tagnum)est = tagnum - 1;

        /* move estimate upward til before range (or 0) */
        while(est > 0 && tags[tagdexes[est]].line >= from)est--;

        /* move estimate downward til tag in or beyond range (or at end) */
        while(est < tagnum && tags[tagdexes[est]].line < from)est++;

        /* show tags from here til out of the range (or off end of list) */
        startdex = est;
        while(est < tagnum){
                tagindex = tagdexes[est++];
                if(tags[tagindex].line > thru)break;
                if(whoshow[tags[tagindex].state])puttag(tagindex);
        }
        return(startdex);       /* return where we started */
}

/* put tag with given index on screen */
/*1*/ puttag(tagindex)
short       tagindex;
{
        UCHAR ch,asciitext[36],tagbuf[36*15];
        short whr,i,xlc,lin,length,trim,lef,top,pixwide,pixhigh;

        /* get tag variables */
        whr     = tags[tagindex].where;
        xlc     = tags[tagindex].xloc;
        lin     = tags[tagindex].line;
```

```
        /* copy tag ascii, limit to 10 characters, typeset it into tagbuf[] */
        length  = 34;
        trim    =  0;   /* how much to trim from final blt */
        for(i=0; i<34; i++){
                asciitext[i] = ch = words[whr++];
                if(ch == '\0'){
                        length = i;
                        break;
                }
        }
        if(length%2){   /* extend to word boundary (but trim final blt) */
                asciitext[length++] = ' ';
                trim = 7;
        }
        asciitext[length] = '\0';
        typeset(asciitext,tagbuf,length,WHITEONBLACK);

        /* put to screen */
        lef     = 192 + 8*xlc;
        pixwide = length*8 - trim;
        top     = yofline(lin) - 8;
        pixhigh = 15;
        solidrect(lef-1, top-1, lef+pixwide+1, top+pixhigh+1, Black);
        outblt(tagbuf, pixwide, pixhigh, lef, top);
        return;
}

                /*2*/ /* MAPPING: TEXTLINE<-->SCREEN */
/* screen y (center if full text) as function of textline */
/*1*/ yofline(textline)
short           textline;
{
        if(textline < topmidlin-2)return(16+textline);      /*above top window*/
        if(textline > botmidlin+2)return(16+150+textline); /*below bot window*/
        if(textline > topmidlin+2 &&
           textline < botmidlin-2)return(16+75+textline);  /* between windows*/
        if(textline < topmidlin+3)return( 54+textline+(textline-topmidlin)*15);
        /* in top--^  in bot-->*/ return(129+textline+(textline-botmidlin)*15);
}

/* determine textline pointed to at screenline y */
/*1*/ lineofy(y)
short           y;
{
        short topoftop,botoftop,topofbot,botofbot;

        /* determine y-levels of transitions */
        topoftop = yofline(topmidlin-3);    botoftop = yofline(topmidlin+3);
        topofbot = yofline(botmidlin-3);    botofbot = yofline(botmidlin+3);

        /* interpret y in terms of textline */
        if(y <= topoftop              )return(y -  16);      /*above top window*/
        if(y >= botofbot              )return(y - 166);      /*below bot window*/
        if(y>=botoftop && y<=topofbot)return(y -  91);      /*between windows */
        if(y<botoftop)return(topmidlin-2+(y-topoftop)/16); /*in top window   */
        else          return(botmidlin-2+(y-topofbot)/16); /*in bot window   */

}
```

I claim:

CLAIMS

1. A system for displaying in an available area of a display screen a larger portion of a document than can be legibly displayed in that area at one time, the system comprising:

(A) a display screen,

(B) greeking means for displaying on the display screen at least a portion of the document in a greeked form in which text is not legible, but that does portray overall shapes in the document,

(C) legible display means for displaying on the display screen a portion of the document in legible form,

(D) drawing control means for directing operation of the greeking means and the legible display means to display a continuous portion of the document as a combination of greeked form and legible form positioned such that the legible form is positioned on the display in its appropriate context relative to the greeked form,

(E) a pointing device for use by a user of the system,

(F) movement control means responsive to the pointing device for directing operation of the drawing control means, the legible display means, and the greeking means to change the portion of the document that is displayed by the legible display means such that a window of legibility is movable through the greeking by the user.

2. The computer system of claim 1 wherein there are tag strings associated with specific locations in the document, and further comprising tag display means for displaying the tag strings on the display screen in legible form at locations on the display screen at which the specific locations in the document are displayed, such

that legible tags are displayable in both greeked and legible portions of the document display.

3. The computer system of claim 1 wherein portions of a document to be displayed have been identified for highlighting and wherein the greeking means and the legible display means each display the identified portion of such document such that, in both greeked and legible portions of the document display, highlighted portions of the document are visually distinguishable from other portions of the document.

4. The computer system of claim 3 wherein the highlighting is effected by displaying different portions of the document in different colors.

5. The computer system of claim 1 wherein the pointing device is a stylus and digitizing tablet, wherein locations on the tablet correspond to locations on the display screen and thereby correspond to portions of the document as currently displayed, and wherein the movement control means is responsive to an action of the stylus touching the tablet to direct the legible display means to display the portion of the document currently corresponding to the touched location on the tablet.

6. The computer system of claim 1 wherein the pointing device is a mouse, further comprising mouse pointer control means for receiving data from the mouse and presenting on the display screen a mouse pointer whose location tracks cumulative mouse movements and can, by its location on the display screen, identify a portion of the displayed document, and wherein the movement control means is responsive to an action of clicking a button on the mouse to direct the legible display means to display the portion of the document identified by the location of the display pointer at the time the button is clicked.

7. The computer system of claim 1 wherein the pointing device is a keyboard.

8. A system for displaying in an available area of a display screen a larger portion of a document than can be legibly displayed in that area at one time, the system comprising:

(A) a display screen

(B) legible display means for displaying on the display screen a portion of the document in legible form,

(C) greeking means for displaying on the display screen a portion of the document in a form that is visually compressed in the vertical direction to the extent that individual characters of the document are not legible, where the width of this greeked display is substantially the same width as the display of the legible display means,

(D) a pointing device for use by a user of the system,

(E) control means responsive to the pointing device for directing operation of the greeking means and the legible display means such that an overall view of the document is displayed to the user in greeked form and, by use of the pointing device, the user can move a window of legibility in the vertical direction through the greeked display.

9. The computer system of claim 8 wherein there are tag strings associated with specific locations in the document, and further comprising tag display means for displaying the tag strings on the display screen in legible form at locations on the display screen at which the specific locations in the document are displayed, such that legible tags are displayable at a plurality of horizontal locations on the display, the horizontal location on the display of a tag being substantially the same whether it is displayed in a greeked portion of the document display or it is displayed in a legible portion of the document display.

10. The computer system of claim 8 wherein portions of a document to be displayed have been identified for highlighting and wherein the greeking means

and the legible display means each display the identified portion of such document such that, in both greeked and legible portions of the document display, highlighted portions of the document are visually distinguishable from other portions of the document.

11. The computer system of claim 8 further comprising second greeking means for displaying on the display screen a portion of the document in a form that is visually compressed in both the horizontal direction and in the vertical direction to the extent that individual characters of the document are not legible.

12. The computer system of claim 11 wherein the control means is operable, in response to an event from the pointing device that indicates a location in the display presented by the second greeking means, thereby indicating a location in the document, to cause the legible display means to display a portion of the document that includes the indicated location in the document.

13. The computer system of claim 11 wherein there are tag strings associated with specific locations in the document, and further comprising tag display means for displaying the tag strings on the display screen in legible form at locations on the display screen at which the specific locations in the document are displayed, such that legible tags are displayable in legible portions of the document display and in portions of the document display presented by both greeking means.

14. A method for controlling the display in an available area on a display screen of a document larger than can be legibly displayed in that area at one time, the display being controlled in response to manipulation of a pointing device by a person viewing the display, the method comprising:

(A) presenting on the display screen at least a portion of the document in a greeked form in which text is not legible, but that does portray overall shapes in the document,

(B) presenting on the display a legible display of a portion of the document, the legible display being positioned on the display in the same relation to the

display of the greeked form as the portion of the document displayed in legible form bears to the portion of the document displayed in greeked form, so that the display of the greeked form provides a context for the display in legible form,

(C) accepting input data from the pointing device resulting from the view's manipulation of the pointing device,

(D) changing the portions of the document displayed in greeked and in legible form in response to the input data.

15. The method of claim 14 wherein the changes in the document display responsive to the input data are such that the viewer can use the pointing device to drag the region of legibility.

16. The method of claim 14 wherein the changes in the document display responsive to the input data are such that the viewer can use the pointing device to point to a portion of the document displayed in greeked form, in response to which that portion of the document is displayed in legible form.

17. The method of claim 14 wherein there are tags associated with specific locations in the document, and further comprising displaying legible forms of the tags at locations on the display screen that correspond to the relative locations of the tags in the document, such that tags are displayed both in areas of greeked display and in areas of legible display.

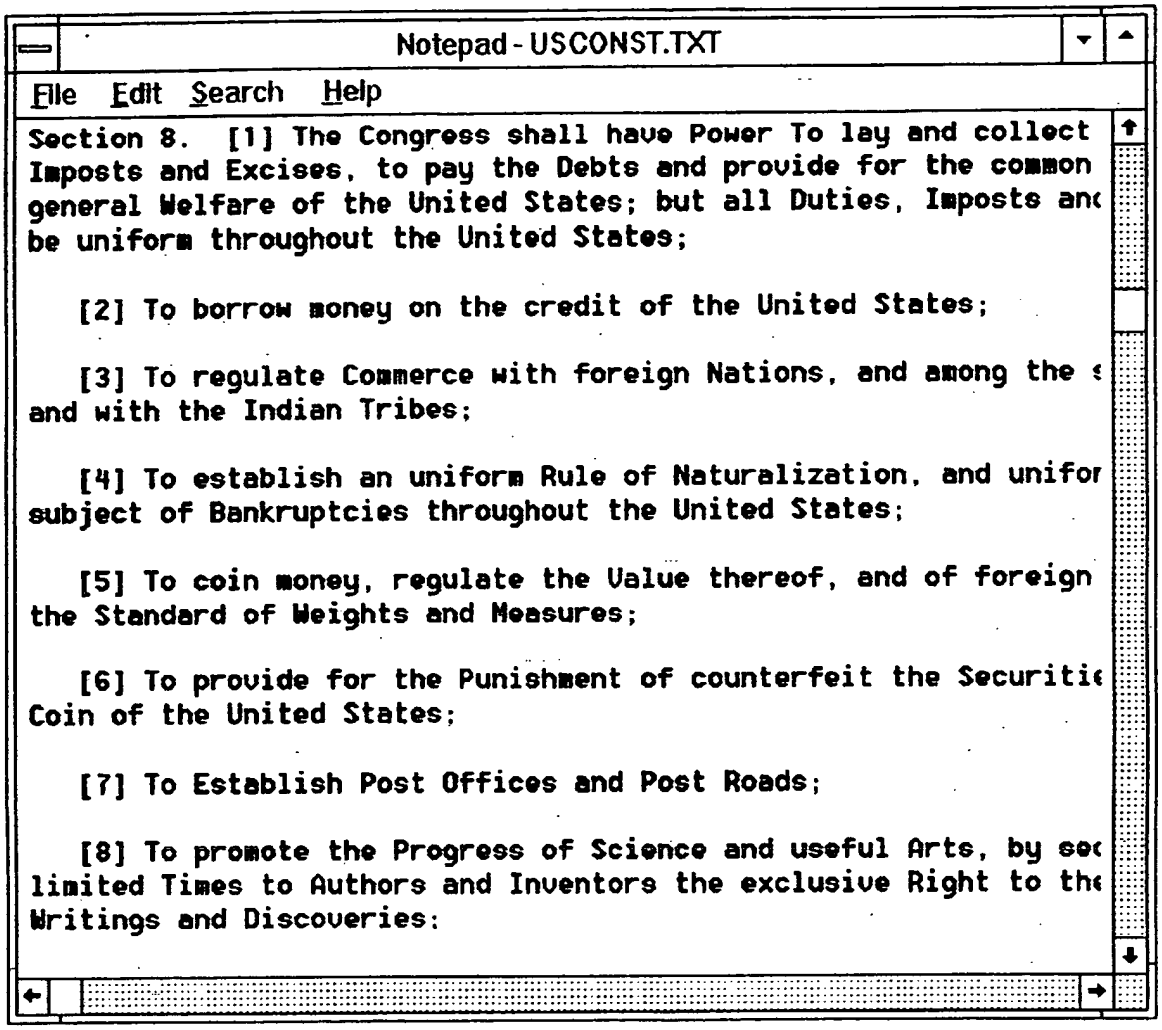

Notepad - USCONST.TXT
File Edit Search Help
Section 8. [1] The Congress shall have Power To lay and collect
Imposts and Excises, to pay the Debts and provide for the common
general Welfare of the United States; but all Duties, Imposts and
be uniform throughout the United States;
[2] To borrow money on the credit of the United States;
[3] To regulate Commerce with foreign Nations, and among the
and with the Indian Tribes;
[4] To establish an uniform Rule of Naturalization, and uniform
subject of Bankruptcies throughout the United States;
[5] To coin money, regulate the Value thereof, and of foreign
the Standard of Weights and Measures;
[6] To provide for the Punishment of counterfeit the Securities
Coin of the United States;
[7] To Establish Post Offices and Post Roads;
[8] To promote the Progress of Science and useful Arts, by sec
limited Times to Authors and Inventors the exclusive Right to the
Writings and Discoveries:

Fig. 1

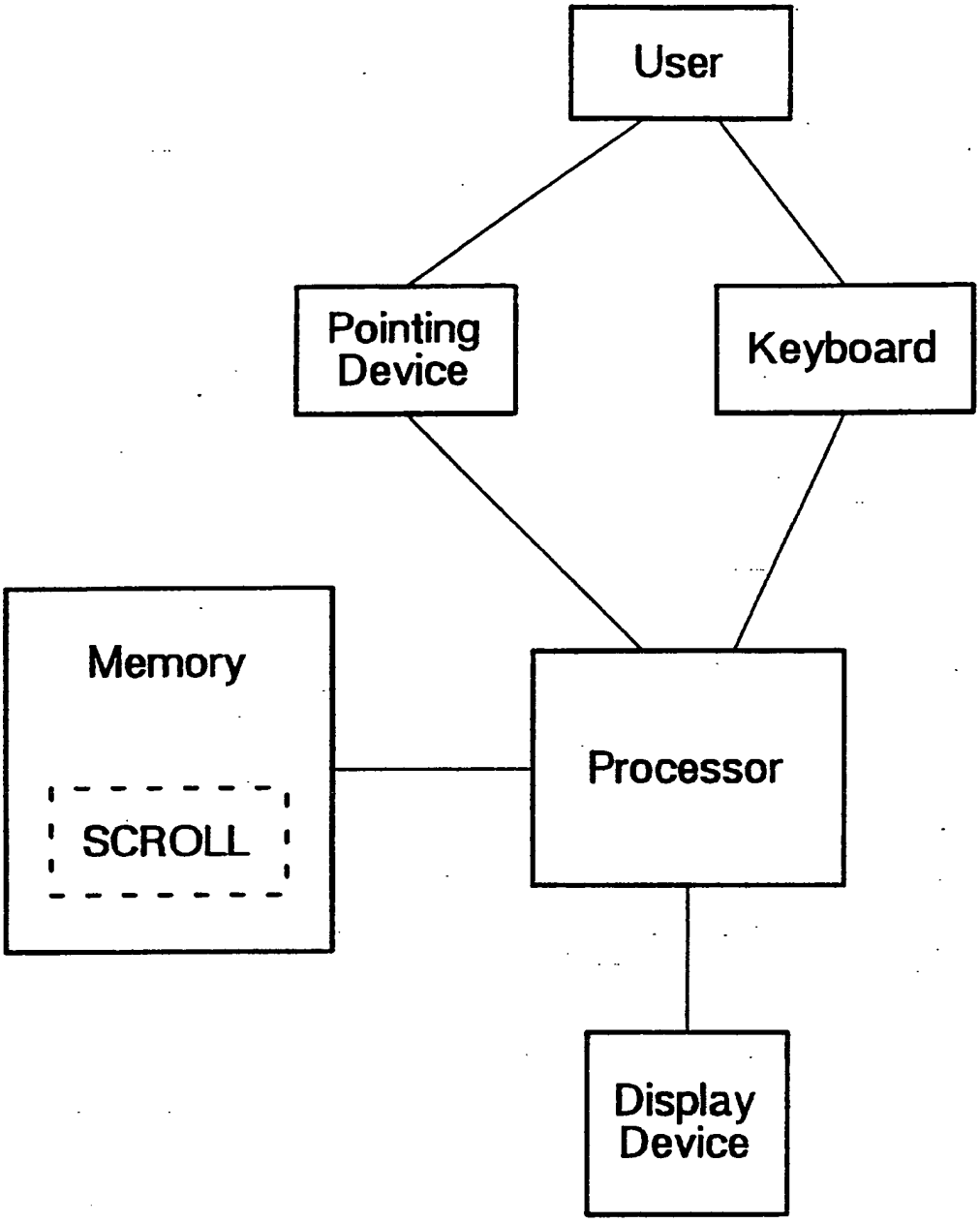

User
Pointing Device
Keyboard
Memory
SCROLL
Processor
Display Device

Fig. 2

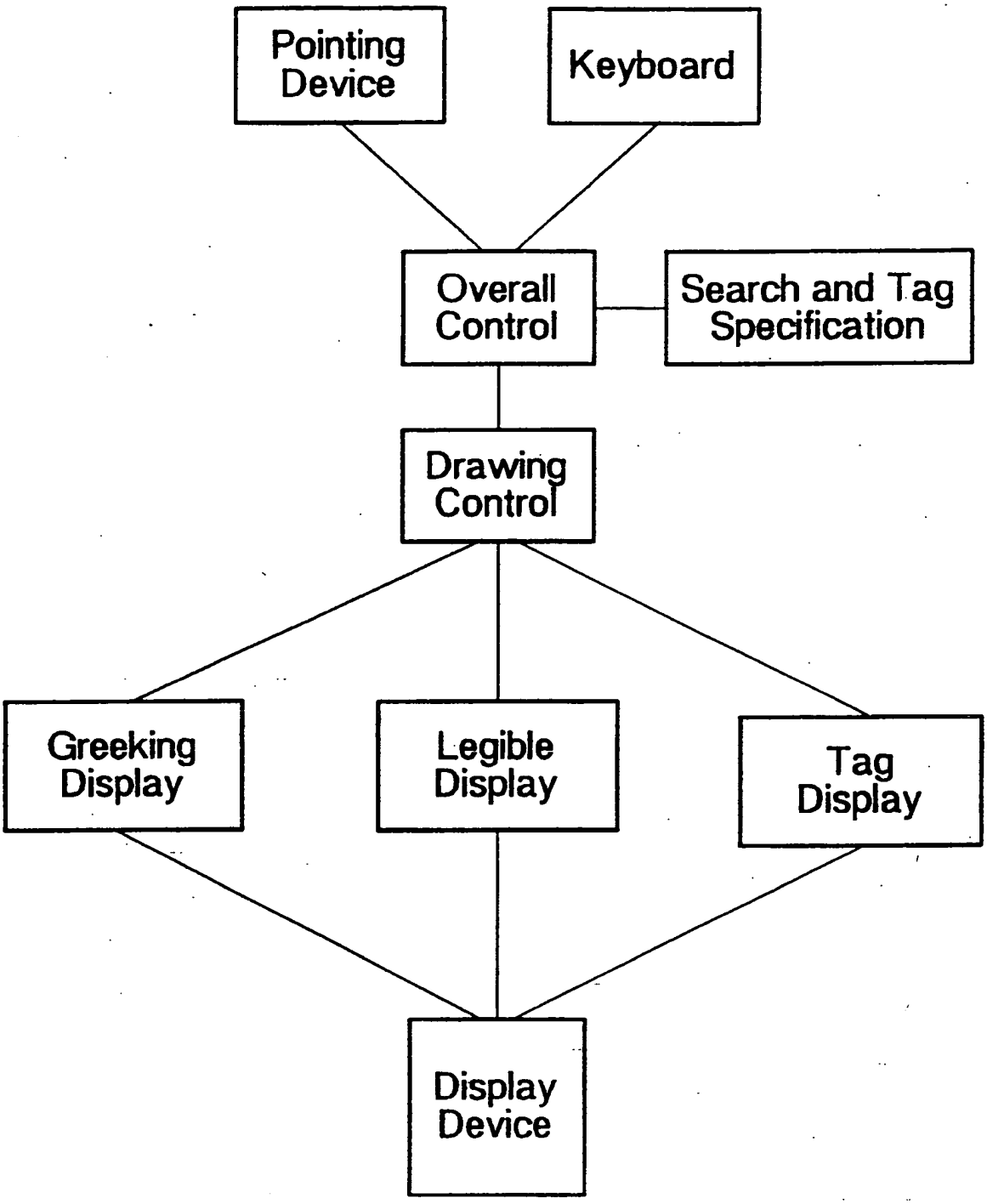
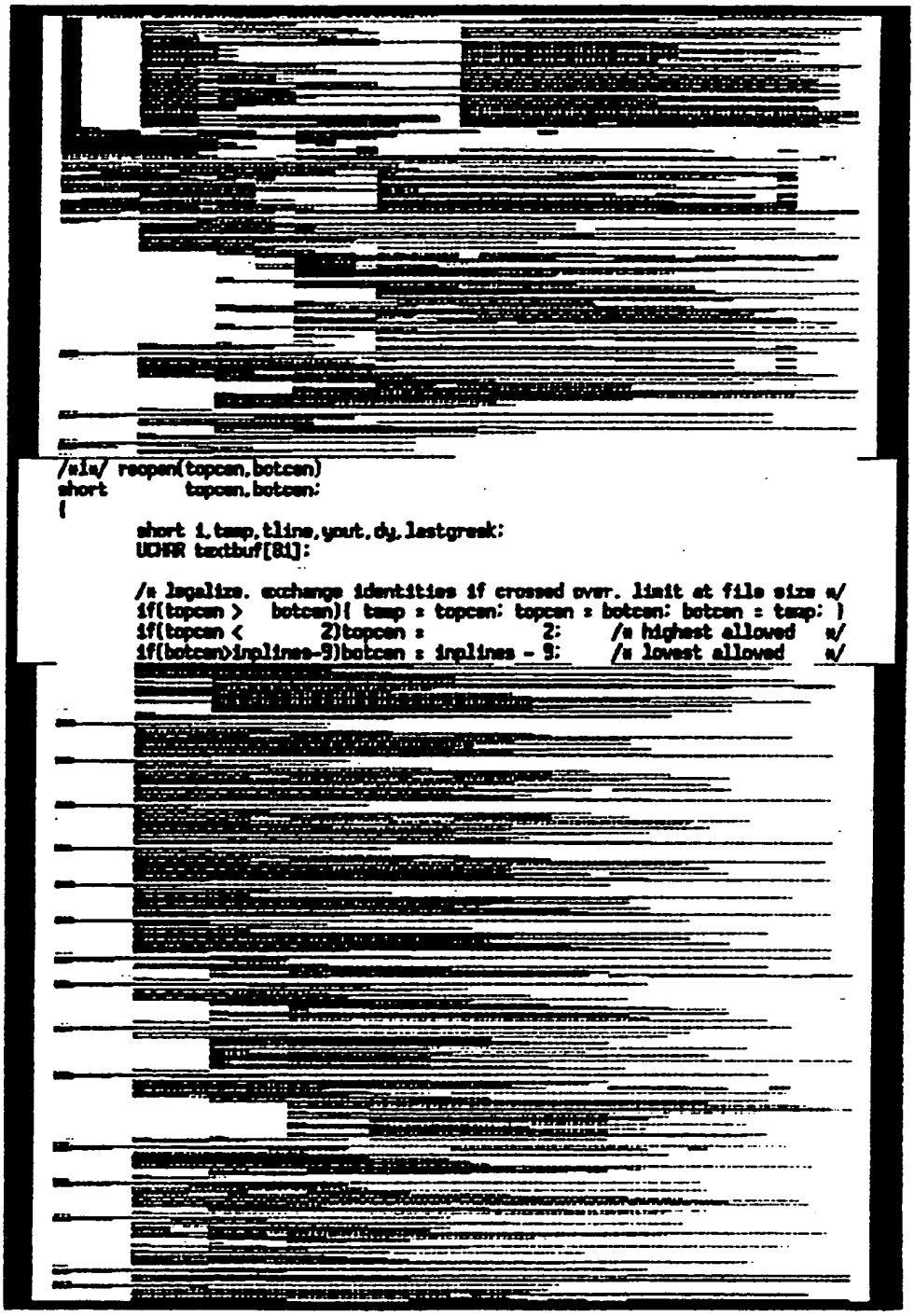

Pointing Device
Keyboard
Overall Control
Search and Tag Specification
Drawing Control
Greeking Display
Legible Display
Tag Display
Display Device

Fig. 3



```
/*ix/ reopen(topcan,botcan)
short       topcan,botcan;
{
        short i,temp,tline,yout,dy,lastgreek;
        UCHAR textbuf[81];

        /* legalize. exchange identities if crossed over. limit at file size */
        if(topcan >  botcan){ temp = topcan; topcan = botcan; botcan = temp; }
        if(topcan <       2)topcan =              2;     /* highest allowed   */
        if(botcan>inplines-9)botcan = inplines - 9;     /* lowest allowed    */
```

Fig. 4

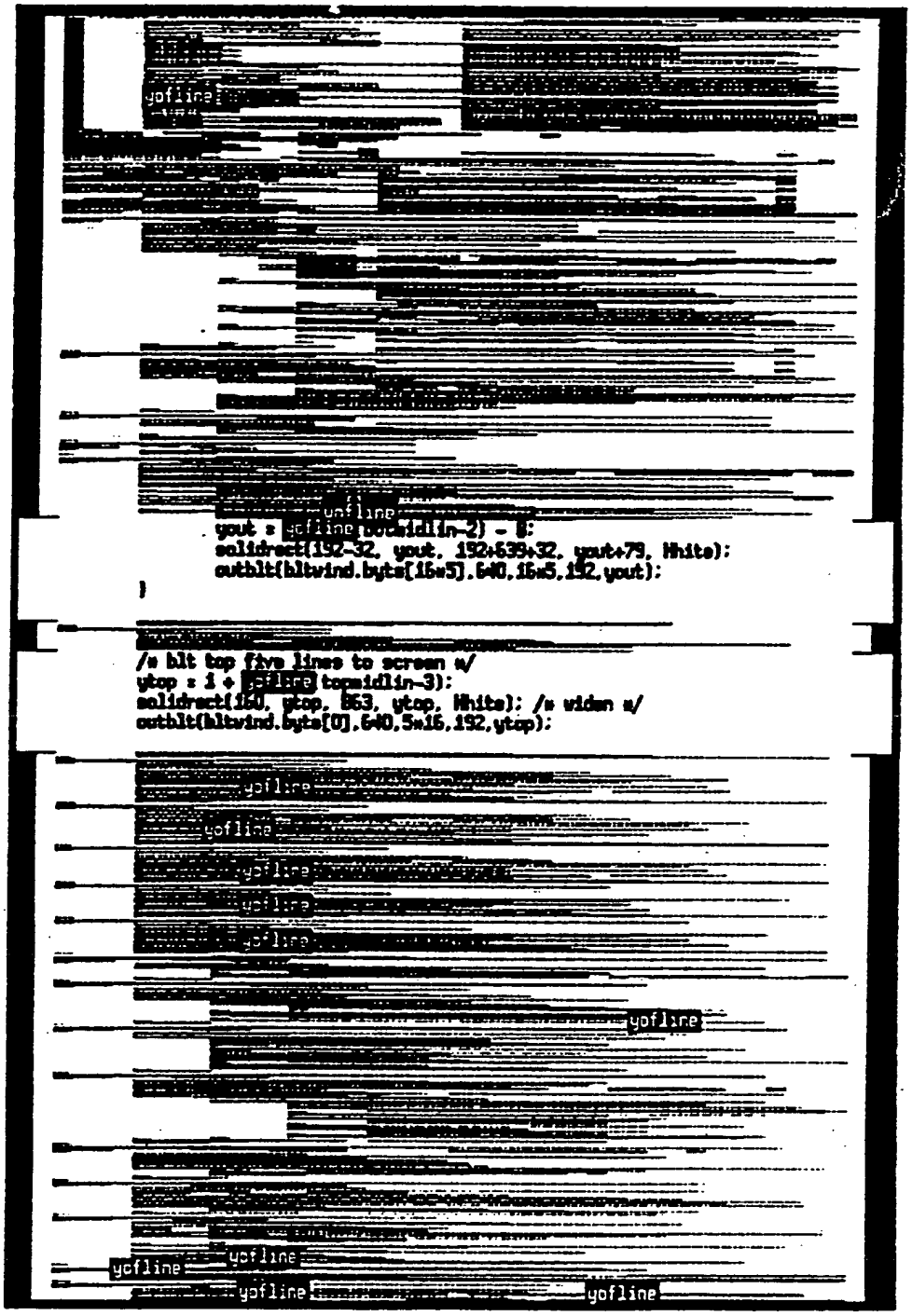

yofline
yofline
yout = yofline(topmidlin-2) - 8;
solidrect(192-32, yout, 192+639+32, yout+79, White);
outblt(hltwind.byts[16*5],640,16*5,192,yout);
}
/* blt top five lines to screen */
ytop = 1 + yofline(topmidlin-3);
solidrect(160, ytop, 863, ytop, White); /* widen */
outblt(hltwind.byts[0],640,5*16,192,ytop);
yofline
yofline
yofline
yofline
yofline
yofline
yofline
yofline
yofline
yofline

Fig. 5

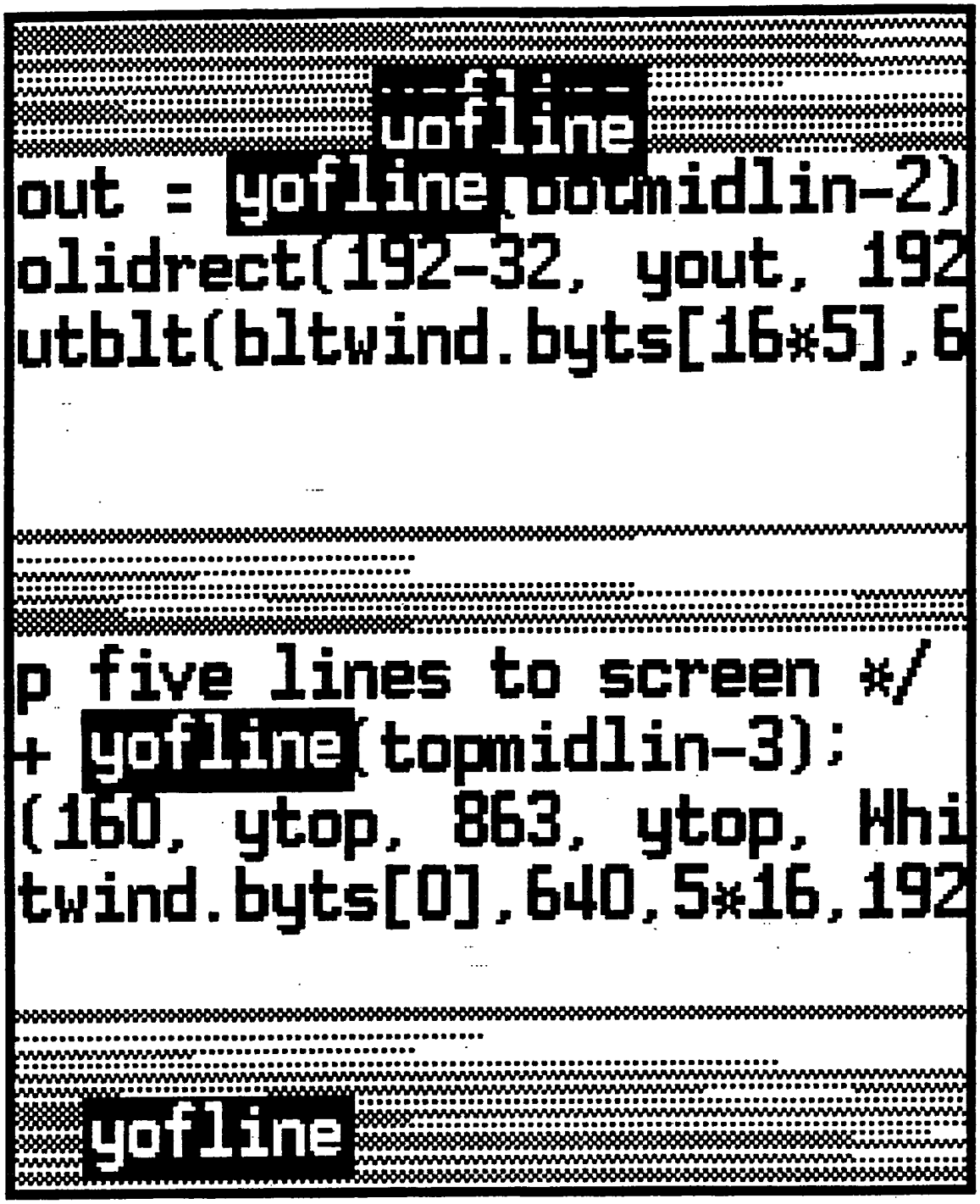

uofline
out = yofline(topmidlin-2)
olidrect(192-32, yout, 192
utblt(bltwind.byts[16*5],6
p five lines to screen */
+ yofline(topmidlin-3);
(160, ytop, 863, ytop, Whi
twind.byts[0],640,5*16,192
yofline

Fig. 6

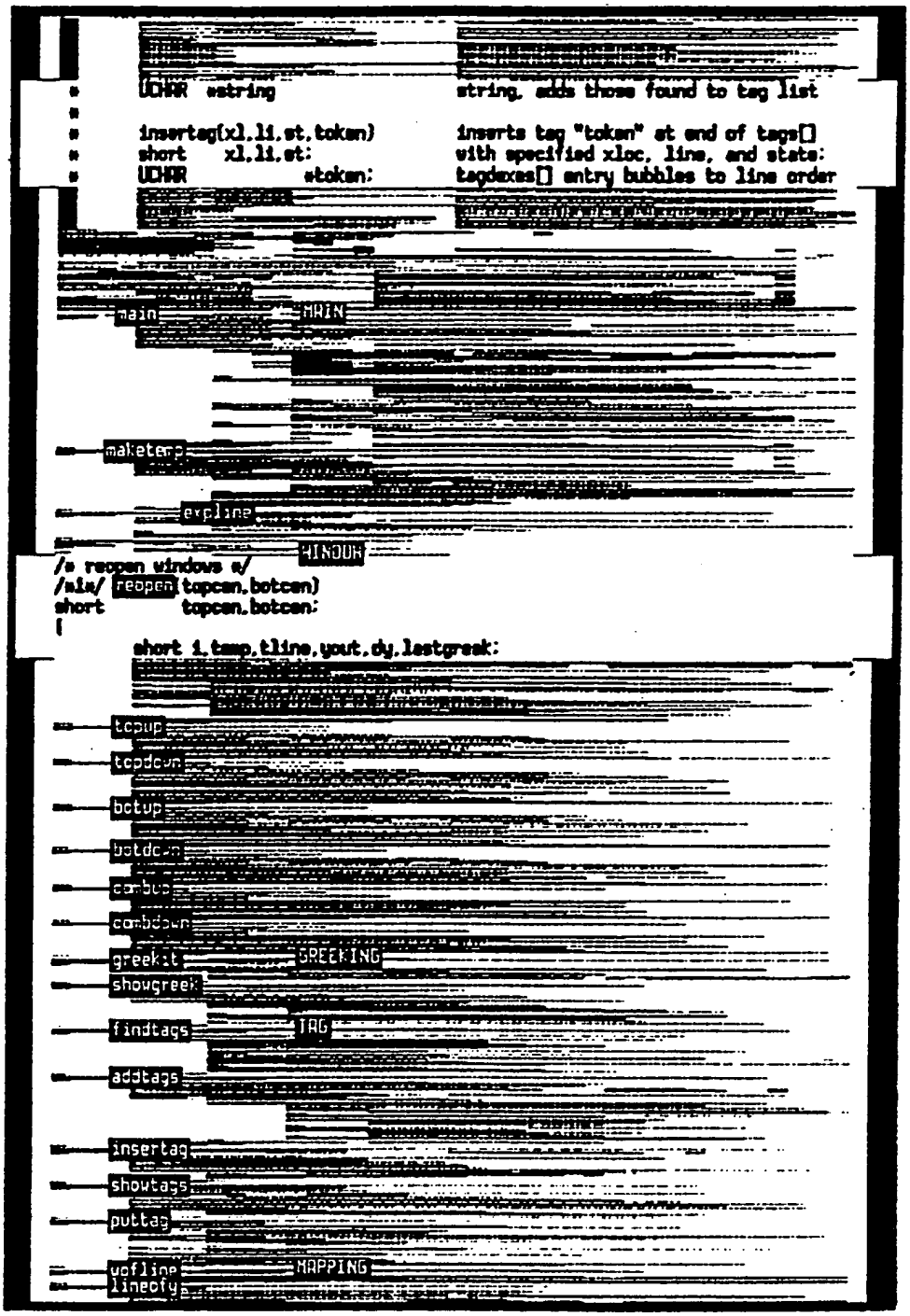

UCHAR *string
string, adds those found to tag list
insertag(xl,li,st,token)
short xl,li,st;
UCHAR *token;
inserts tag "token" at end of tags[]
with specified xloc, line, and state;
tagdexes[] entry bubbles to line order
main
MAIN
maketerp
expline
MINDOW
/* reopen windows */
/win/ reopen(topcan,botcan)
short topcan,botcan;
short i,temp,tline,yout,dy,lastgreek;
topup
topdown
botup
botdown
combup
combdown
greekit
GREEKING
showgreek
findtags
TAG
addtags
insertag
showtags
puttag
vofline
MAPPING
lineofy

Fig. 7

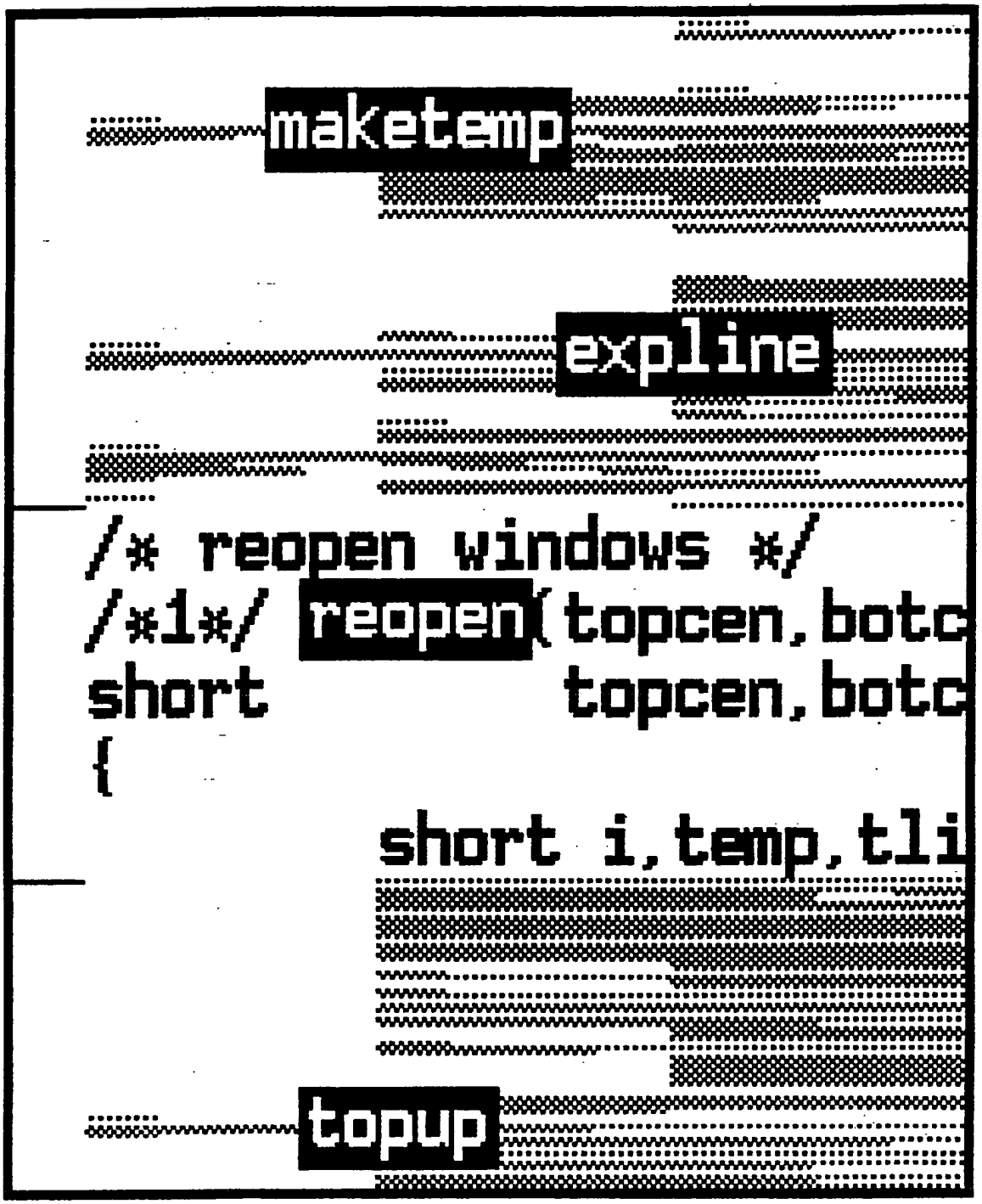

maketemp
expline
/* reopen windows */
/*1*/ reopen(topcen,botc
short topcen,botc
{
short i,temp,tli
topup

Fig. 8

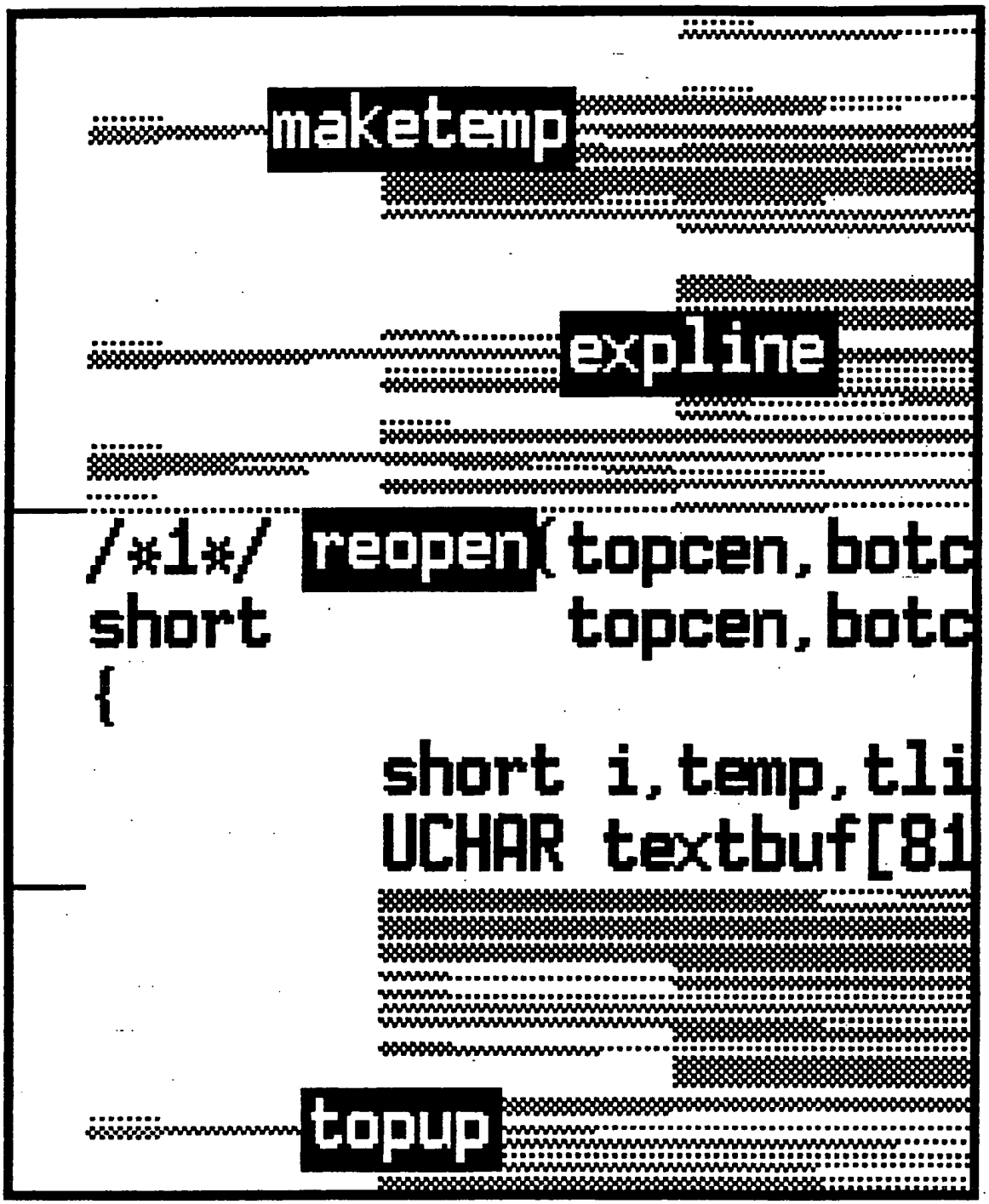

maketemp
expline
/*1*/ reopen(topcen,botc
short topcen,botc
{
short i,temp,tli
UCHAR textbuf[81
topup

Fig. 9

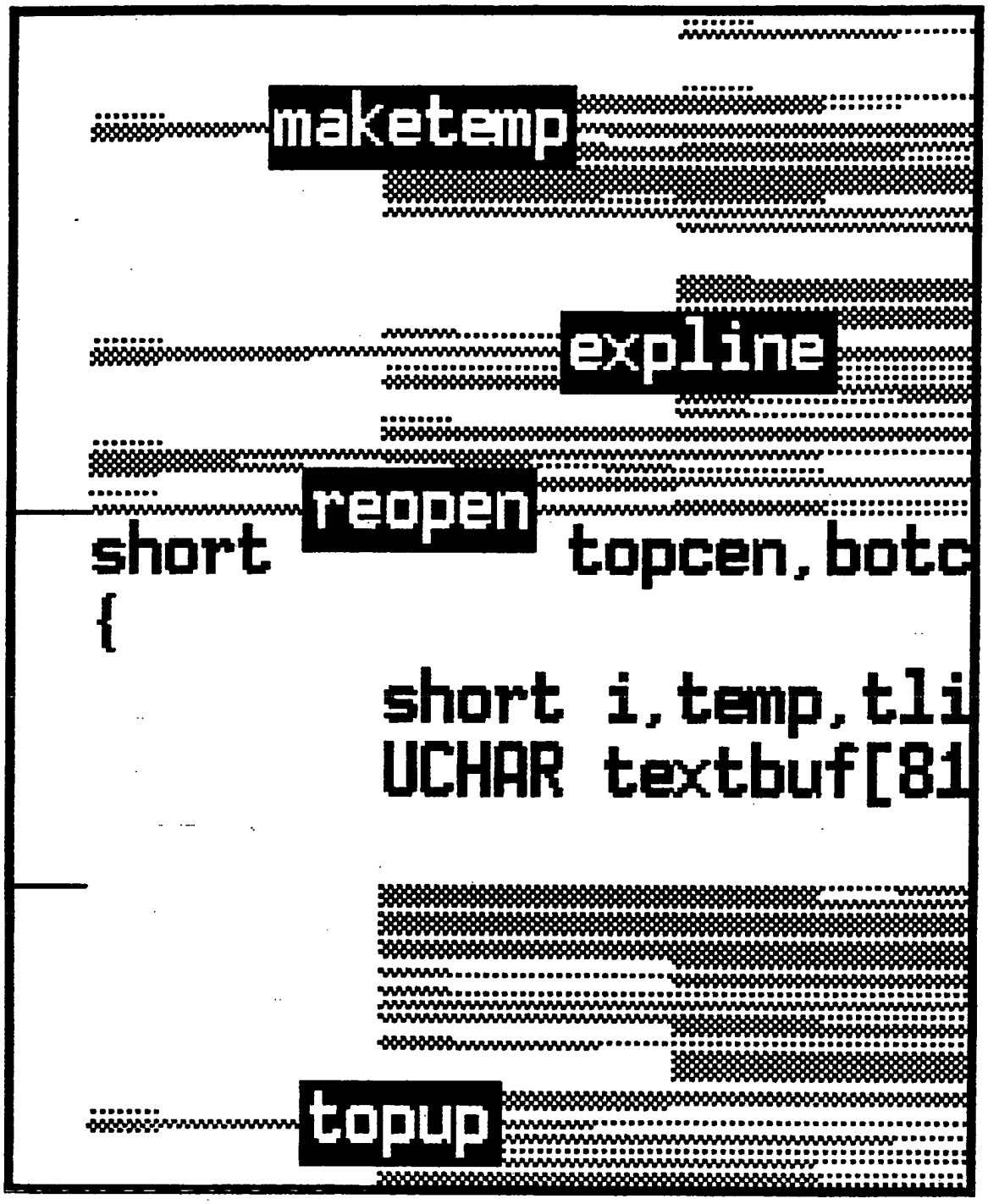

maketemp
expline
reopen
short topcen,botc
{
short i,temp,tli
UCHAR textbuf[81
topup

Fig. 10

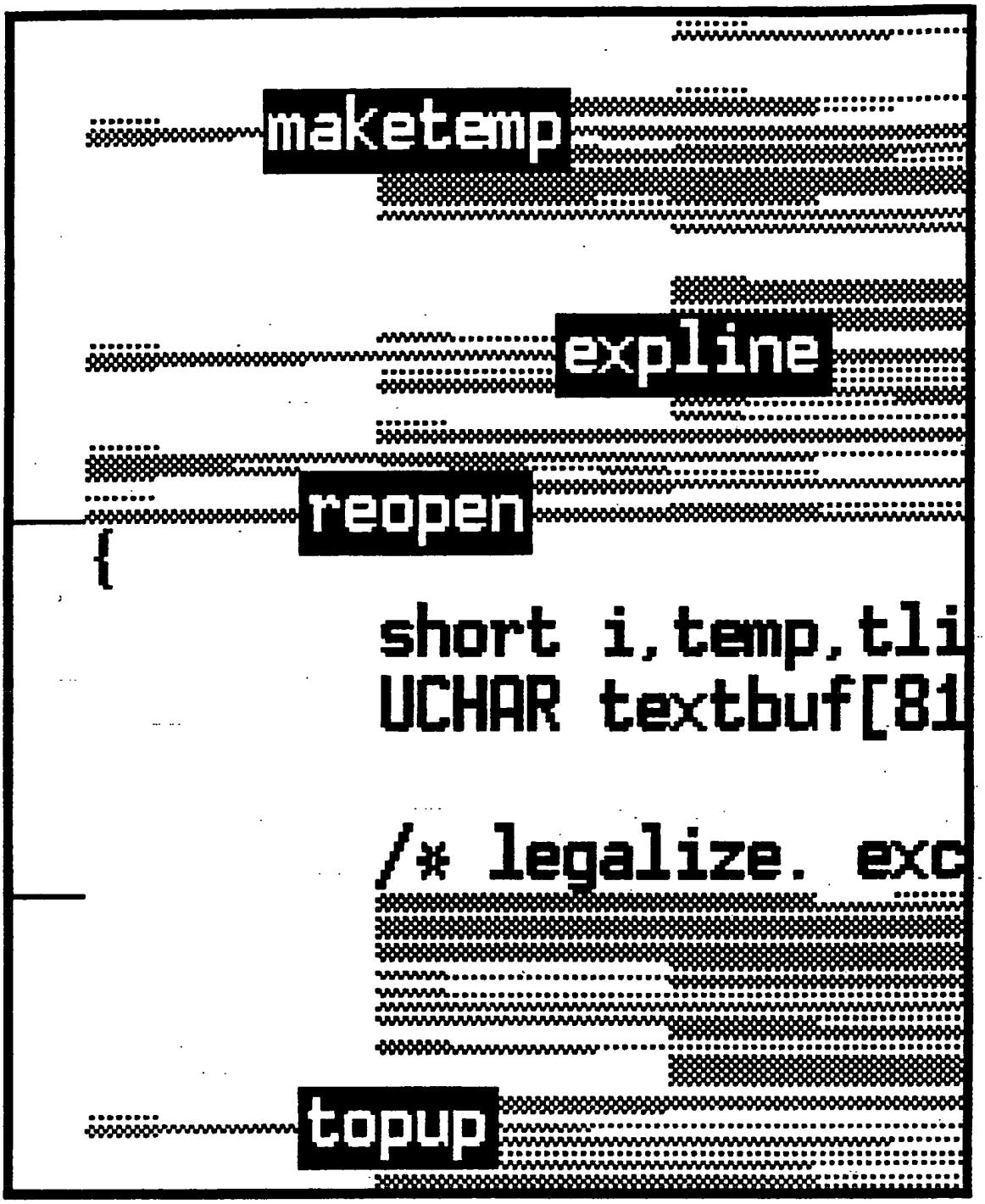

maketemp
expline
reopen
{
short i,temp,tli
UCHAR textbuf[81
/* legalize. exc
topup

Fig. 11

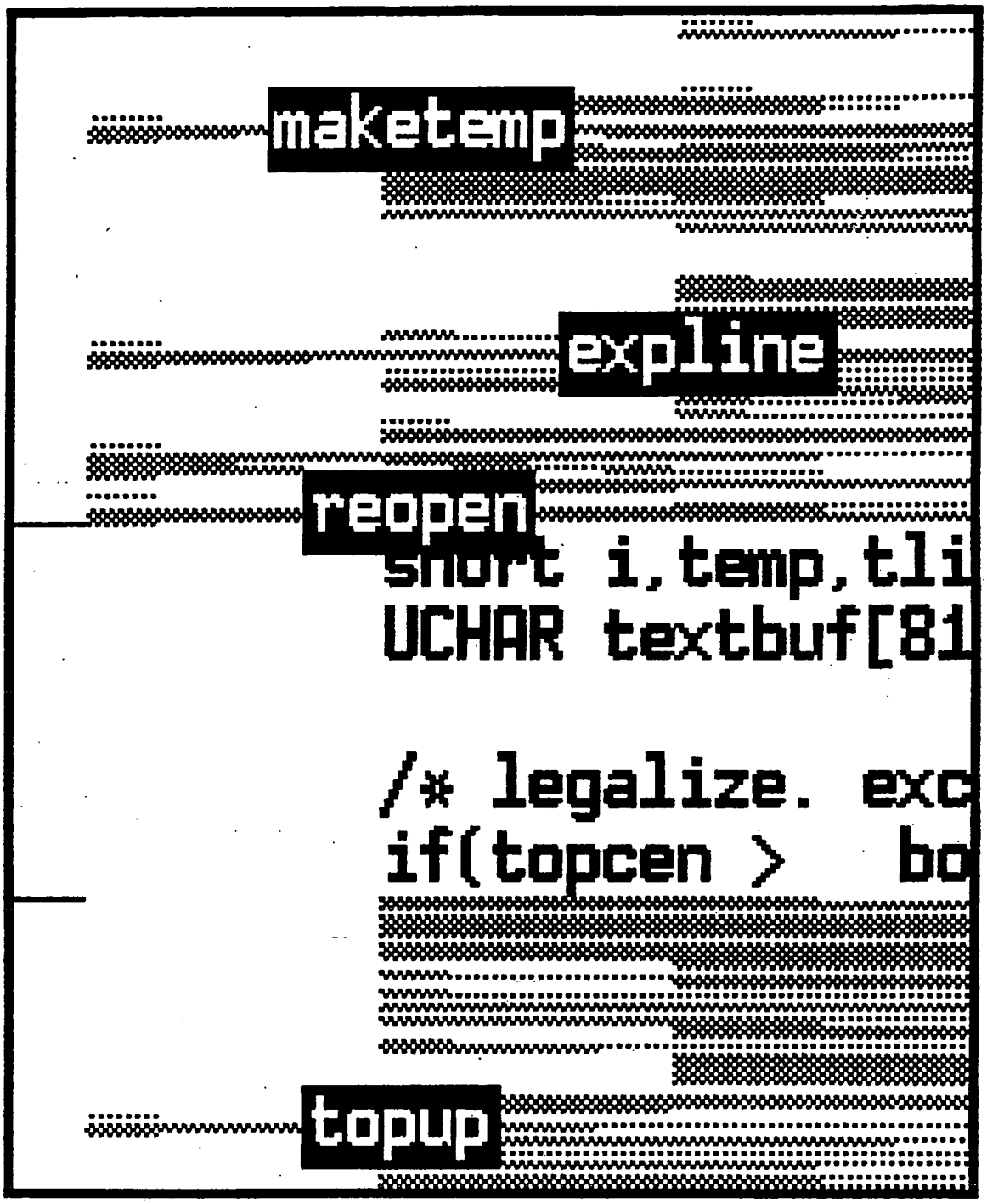

maketemp
expline
reopen
short i,temp,tli
UCHAR textbuf[81
/* legalize. exc
if(topcen > bo
topup

Fig. 12

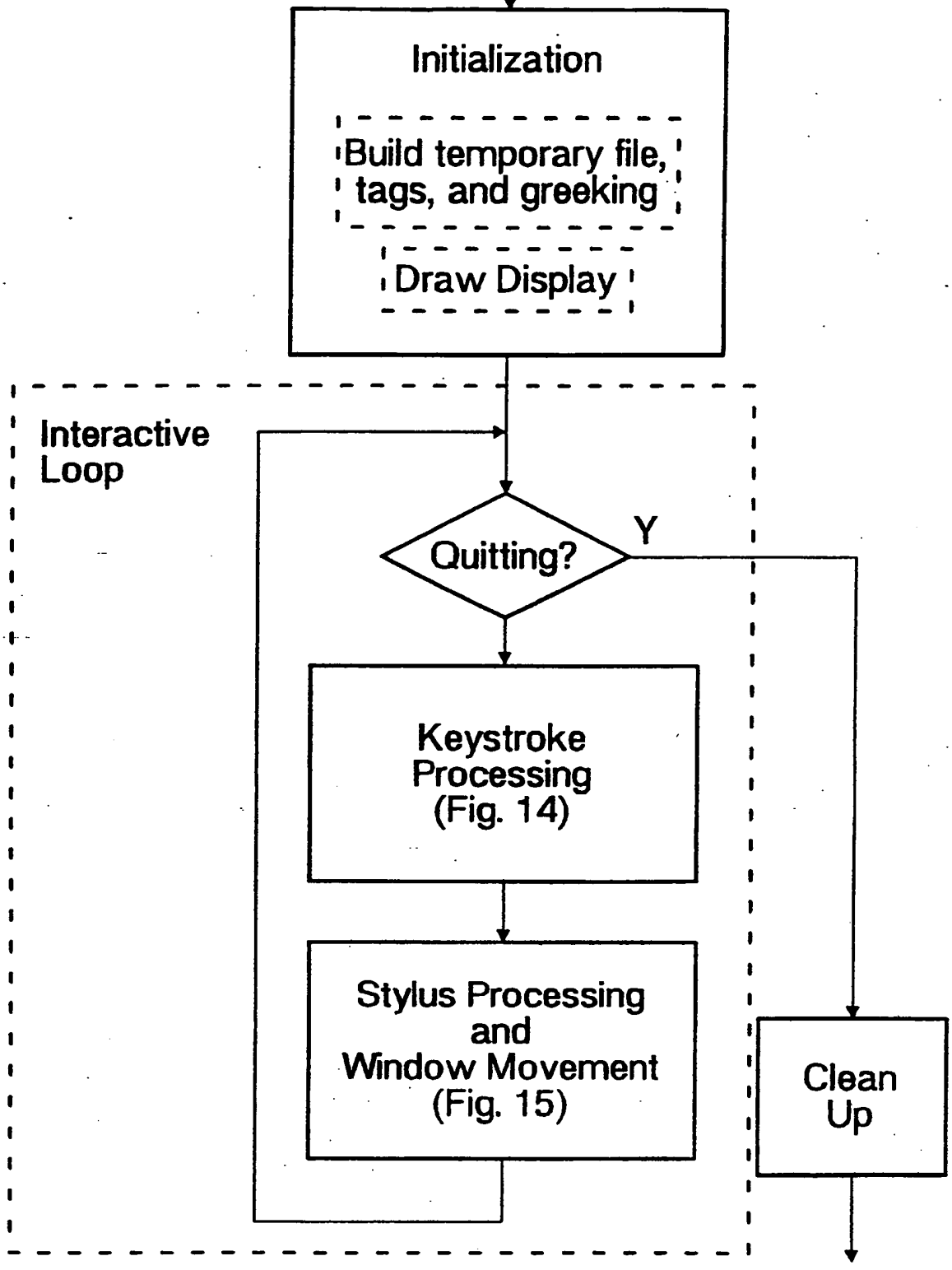

Initialization
Build temporary file, tags, and greeking
Draw Display
Interactive Loop
Quitting?
Y
Keystroke Processing (Fig. 14)
Stylus Processing and Window Movement (Fig. 15)
Clean Up

Fig. 13

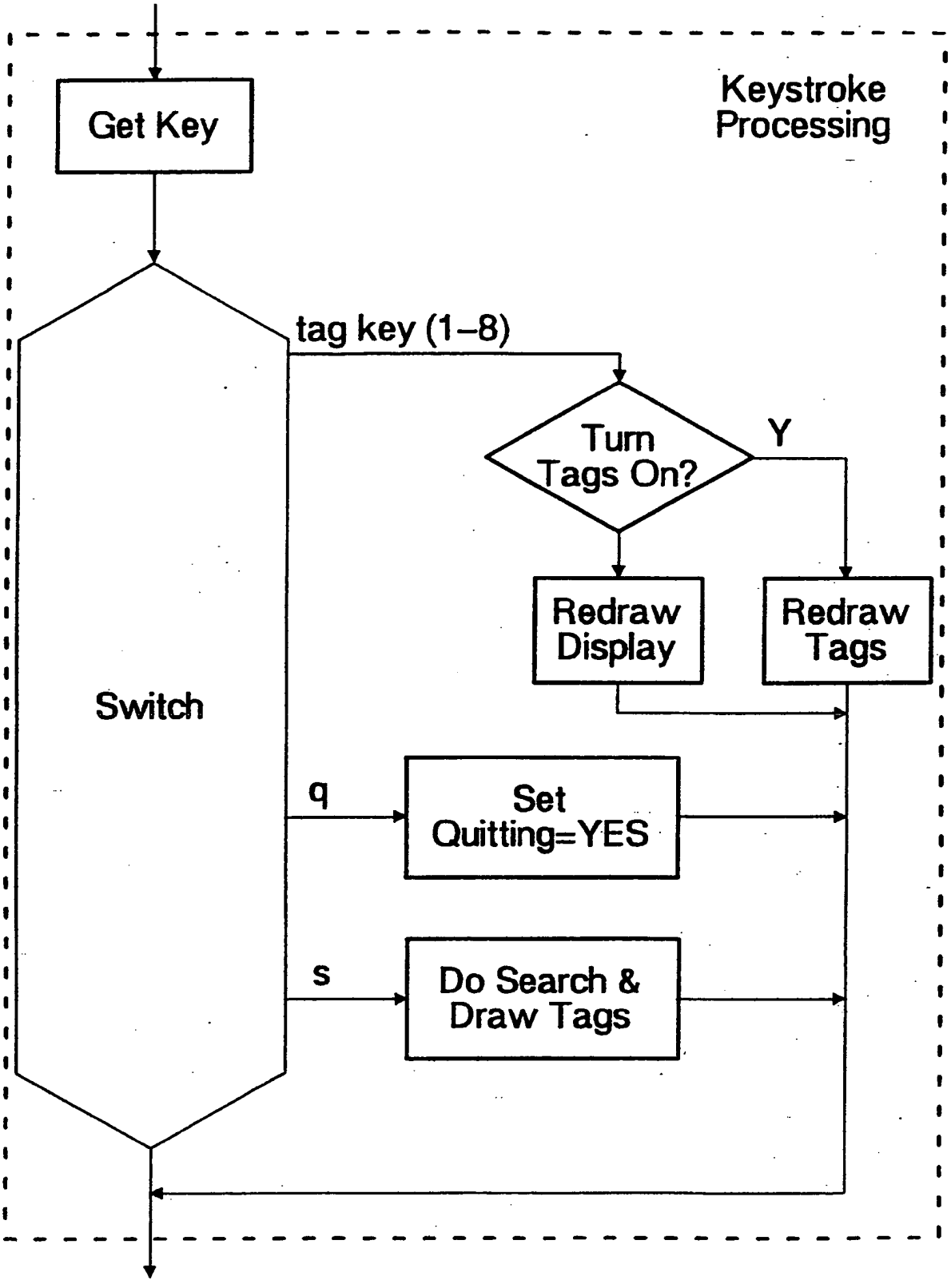

Keystroke Processing
Get Key
Switch
tag key (1–8)
Turn Tags On?
Y
Redraw Display
Redraw Tags
q
Set Quitting=YES
s
Do Search & Draw Tags

Fig. 14

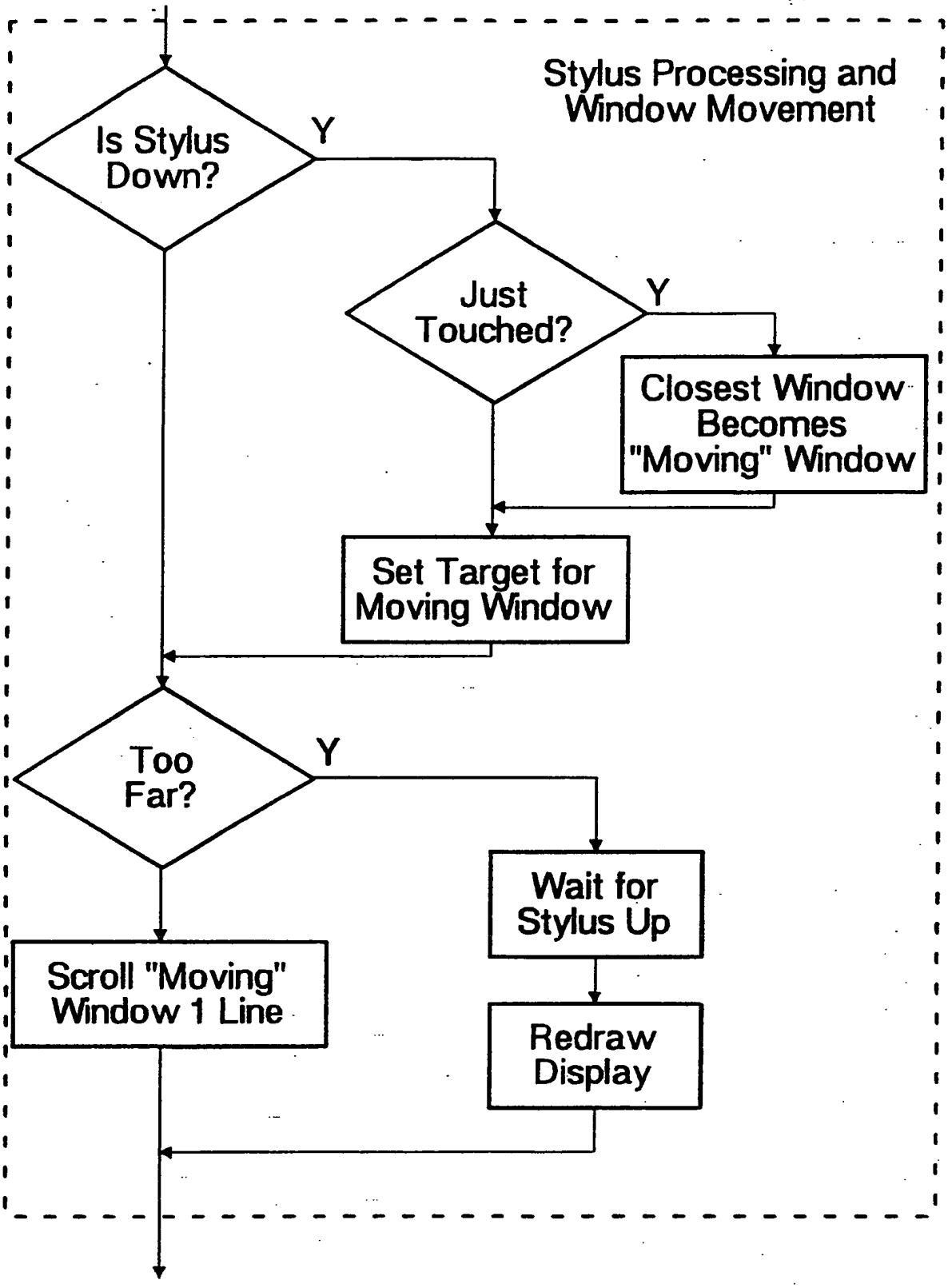

Stylus Processing and Window Movement
Is Stylus Down?
Y
Just Touched?
Y
Closest Window Becomes "Moving" Window
Set Target for Moving Window
Too Far?
Y
Wait for Stylus Up
Scroll "Moving" Window 1 Line
Redraw Display

Fig. 15

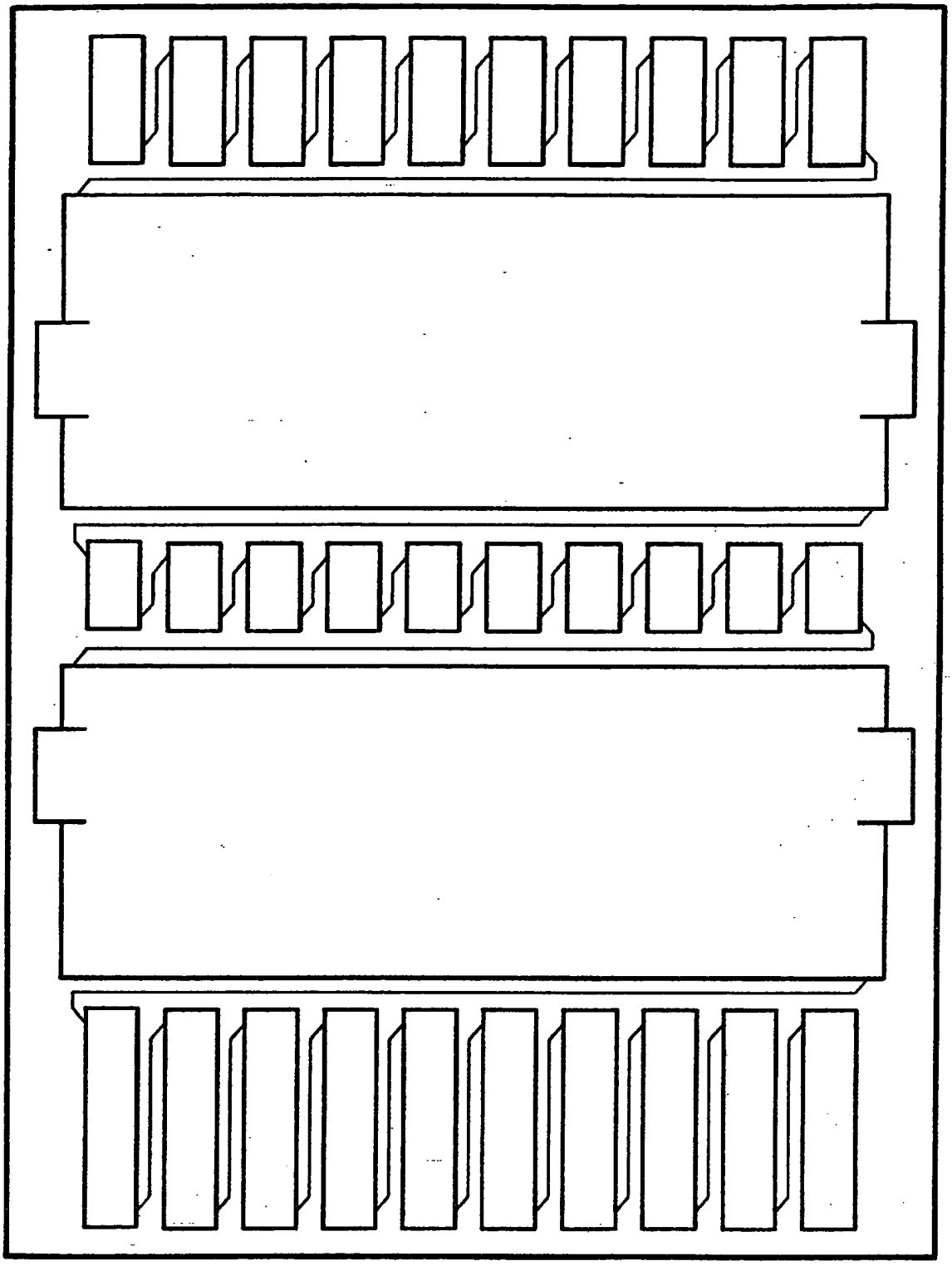


Fig. 16

INTERNATIONAL SEARCH REPORT

International Application No PCT/US 91/01652

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all)[6]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int.Cl. 5 G06F15/20 ; G09G1/00

II. FIELDS SEARCHED

Minimum Documentation Searched[7]

Classification System	Classification Symbols
Int.Cl. 5	G06F ; G09G

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched[8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT[9]

Category °	Citation of Document, [11] with indication, where appropriate, of the relevant passages [12]	Relevant to Claim No.[13]
A	US,A,4 451 900 (S.T. MAYER) May 29, 1984 see column 2, line 18 - column 7, line 34; figure 4	1,8,14
A	GB,A,2 137 788 (GRAVILAN COMPUTER CORP.) October 10, 1984 see page 1, line 20 - page 4, line 23; figures 1-7	1,8,14
A	US,A,4 823 303 (M. TERASAWA) April 18, 1989 see column 1, line 44 - column 4, line 24; figures 1-5	1,8,14
A	US,A,4 428 065 (W.S. DUVALL) January 24, 1984 see column 1, line 55 - column 2, line 48	1,8,14
A	GB,A,2 156 118 (CANON) October 2, 1985 see page 1, line 51 - page 2, line 116; figures 1-5	1,8,14

° Special categories of cited documents : [10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
09 SEPTEMBER 1991	9. 10. 91
International Searching Authority	**Signature of Authorized Officer**
EUROPEAN PATENT OFFICE	Patricia Smith P.h. Smith

ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO.

US 9101652
SA 45958

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

23/09/91

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US-A-4451900	29-05-84	None	
GB-A-2137788	10-10-84	DE-A- 3413130	11-10-84
		FR-A- 2544114	12-10-84
		JP-A- 60006994	14-01-85
US-A-4823303	18-04-89	JP-A- 63024419	01-02-88
US-A-4428065	24-01-84	JP-A- 56007175	24-01-81
GB-A-2156118	02-10-85	JP-A- 60176090	10-09-85
		DE-A,C 3506321	29-08-85
		US-A- 4881064	14-11-89

EPO FORM P0479

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☑ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☑ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.